Physikalisch-Technische Bundesanstalt

DECKBLATT

Projekt	PSP-Element	Obj. Kenn.	Funktion	Komponente	Baugruppe	Aufgabe	UA	Lfd. Nr.	Rev
N A A N	N N N N N N N N N N	N N N N N N	N N A A A N N	A A N N N A	A A N N	X A A X X	A A	N N N N	N N
9K	352126.39	---	---	---	---	EGA	RE	0001	00

Titel der Unterlage: Modellrechnungen zur Grundwasserbewegung mit dem Programm FEM 301; Nagra, Interner Bericht 88-63; Particle Tracking Advektiver Transport im Grundwasser Theorie und Code-Beschreibung des Programms Track, [redacted] Zürich, März 1989 (lfd. Nr. 285) Ersetzt EU 236

Seite: I.

Stand: März 89

Ersteller: NAGRA

Textnummer:

Stempelfeld

PSP-Element TP 2: 9K/2122423

zu Plan-Kapitel: 3.1.10.4

		PL	PL
		[redacted]	
		06.89	06.89
		Freigabe für Behörden	Freigabe im Projekt

REVISIONSBLATT

Projekt	PSP-Element	Obj. Kenn.	Funktion	Komponente	Baugruppe	Aufgabe	UA	Lfd. Nr.	Rev
N A A N	N N N N N N N N N	N N N N N N	N N A A A N N	A A N N N A	A A N N	X A A X X	A A	N N N N	N N
9K	352126.39	---	---	---	--- EGA		RE	0001	00

Titel der Unterlage: Modellrechnungen zur Grundwasserbewegung mit dem Programm FEM 301; Nagra, Interner Bericht 88-63; Particle Tracking Advektiver Transport im Grundwasser Theorie und Code-Beschreibung des Programms Track, [redacted] Zürich, März 1989 (lfd. Nr. 285) Ersetzt EU 236

Seite: II.

Stand: März 89

Rev.	Revisionsst. Datum	verant. Stelle	Gegenzeichn. Name	rev. Seite	Kat. *)	Erläuterung der Revision

*) Kategorie R = redaktionelle Korrektur
Kategorie V = verdeutlichende Verbesserung
Kategorie S = substantielle Änderung
Mindestens bei der Kategorie S müssen Erläuterungen angegeben werden.

V 86 / 718 / 2

Nagra
Nationale Genossenschaft für die Lagerung radioaktiver Abfälle

Cédra
Société coopérative nationale pour l'entreposage de déchets radioactifs

Cisra
Società cooperativa nazionale per l'immagazzinamento di scorie radioattive

INTERNER BERICHT 88-63

PARTICLE TRACKING
ADVEKTIVER TRANSPORT IM GRUNDWASSER
Theorie und Code-Beschreibung des Programmes TRACK

von [redacted] VAW ETH Zürich

MÄRZ 1989

STICHWÖRTER

Geschwindigkeitsfeld, Algorithmus, FEM, Kontinuitätsgleichung, Stromlinien, Software, Modellierungsprogramm

[redacted]

VERSUCHSANSTALT FÜR WASSERBAU, HYDROLOGIE UND GLAZIOLOGIE
DER
EIDGENÖSSISCHEN TECHNISCHEN HOCHSCHULE ZÜRICH

PARTICLE TRACKING - ADVEKTIVER TRANSPORT IM GRUNDWASSER

THEORIE UND CODE-BESCHREIBUNG DES PROGRAMMS TRACK

ZURICH, MARCH 1989

INHALTSVERZEICHNIS

ZUSAMMENFASSUNG

Vorliegender Bericht beschreibt den Programm-Code TRACK. TRACK berechnet Fliesswege und -zeiten im dreidimensionalen, gesättigten, stationären Grundwasserträger auf der Basis der Potentialverteilung. Das Potential wird mit dem Code FEM301 berechnet. Es wird die Trajektoriendifferentialgleichung gelöst. Eine entscheidende numerische Vereinfachung ist die Formulierung der Differentialgleichung im transformierten lokalen Koordinatensystem.

1. EINLEITUNG

1.1 Voraussetzungen

Das Programm TRACK berechnet Trajektorien und Fliesszeiten entlang Stromlinien in einem stationären, gesättigten, dreidimensionalen Grundwasserträger. Grundlage ist der Code FEM301 (beschrieben in [redacted] 1985), der in demselben Aquifer die Potentialverteilung mit der Methode der finiten Elemente bestimmt.

Die Kenntnis der Trajektorien ist wichtig, um Aussagen über den Verlauf von Partikeln von Endlagern in die Biosphäre machen zu können. Daraus können Schlüsse gezogen werden, welcher Art der Chemismus sein wird, da Aufenthaltszeiten in den verschiedenen geologischen Schichtungen von entscheidender Bedeutung sind.

Es wird dieselbe Diskretisierung in finite Elemente benützt, wie in FEM301. Das Einlesen der Elementdaten, bestehend aus Elementkonfiguration, Knotenkoordinaten und Gebietsparameter, geschieht durch FEM301-Routinen. Ebenfalls werden das die Potentiale enthaltende Resultatfile und ein neu zu kreierendes Startkonditionenfile benötigt. Nach dem Einlesen der Eingabefiles werden für jeden Startpunkt Trajektorien berechnet.

Der vorliegende Bericht beinhaltet vier zentrale Teile:
Im ersten Teil werden physikalische und mathematische Grundlagen und numerische Methoden bereitgestellt.
Im zweiten Teil wird der Aufbau des Programmcodes beschrieben. Schrittweise werden theoretische Konzepte zu einer numerischen Lösung umgesetzt und jeweils die verwendeten Methoden erwähnt oder dargestellt. Abschliessend folgt eine Zusammenstellung der grundlegenden Formeln.
Der dritte Teil kann als "Handbuch" für den Benützer betrachtet werden. Eingabe und Ausgabe sind mit Beispielen dokumentiert.
Der vierte Teil liefert ein analytisches Testbeispiel zur Code-Verifikation.
Des weiteren ist im Anhang ein aktuelles, datiertes Listing abgedruckt.

1.2 Modellannahmen

Der dreidimensionale Grundwasserträger wird als poröses Medium betrachtet. Geklüftete Gebiete können als äquivalent poröses Medium definiert werden. Das Gesetz von Darcy ergibt mit der Kontinuitätsgleichung die zu lösenden Differentialgleichungen.

In FEM301 basiert die Lösungsmethode auf dem finite-Elemente-Verfahren nach [redacted]. Es werden quadratische

Interpolationspolynome benützt. Klüfte, Stollen etc. können durch zwei- und eindimensionale Elemente diskretisiert werden. Permeabilitäten und Porositäten sind elementweise konstant.

Zur Berechnung der Trajektorien kann der Modellinput belassen werden. Die Behandlung von zwei- und eindimensionalen Elementen erfordert ein konzeptuelles Vorgehen. Es wurde ein Konzept gewählt, in welchem die konservative Philosophie des schnellsten Weges verfolgt wird. Nach demselben Konzept werden auch Schwierigkeiten behandelt, die sich bei stark kontrastierenden Elementübergängen ("Sprünge" in der Permeabilität) ergeben (siehe Kapitel 3.2.6 Oszillationen).

1.3 Anwendungen und Status des Codes

Der Vergleich mit analytischen Beispielen zeigt gute Uebereinstimmung der numerischen Resultate. Schwierigkeiten mit "pinched sided elements" (Elemente, wo Kanten zusammengeklappt werden oder zu Knoten degenerieren) sind noch nicht vollständig aus dem Weg geräumt. In diesem Zusammenhang ist die Mitarbeit von Benützerseite unerlässlich.

In der vorliegenden Version des Codes ist es möglich, Modelle mit reduzierter Raumdimension zu behandeln. Der Aufwand für die Anpassung des Codes an andere Programme ist abhängig davon, mit welchen Elementtypen diskretisiert wird. Falls völlig fremde Elementtypen verwendet werden, sind einige Aenderungen notwendig, da gewisse Programmbausteine elementtopologische Eigenschaften berechnen müssen.

Der Stand des Programmes ist nicht endgültig. Vorgesehen ist, dass zwecks Qualitätssicherung das Programm nochmals überarbeitet und leserlich gestaltet wird. Ebenfalls sind einige, die Lesbarkeit fördernde Aenderungen am Programmaufbau vorgesehen.

1.4 Verdankungen

Der Autor des Berichtes möchte die Mitarbeit von [redacted] (Motor Columbus, Ing., Baden) nicht unerwähnt lassen. Er war massgeblich an der Entwicklung des Programmkonzeptes beteiligt, hat einige Programmteile neu geschrieben und bedeutende Fehler im Code entdeckt und korrigiert. Entscheidende Hinweise und Hilfe, besonders in der Startphase des Programms, kamen von [redacted] (VAW-ETH, Zürich). Ebenfalls hat er den vorliegenden Text korrigiert. [redacted] und [redacted] (VAW-ETH, Zürich) verdankt der Autor die Mithilfe beim Austesten des Programms, indem sie beim Erstellen des Netzes und der Eingabedaten Hilfe geleistet haben.

2. ZUSAMMENFASSUNG DER THEORIE (vgl. [redacted] 1985)

Es wird ein dreidimensionaler gesättigter Grundwasserträger betrachtet, dessen poröse Matrix stationär durchströmt wird. Im Falle von geklüfteten Systemen kann ein äquivalent poröses Medium definiert werden. Das Gesetz von Darcy ergibt zusammen mit der Kontinuitätsgleichung einen vollständigen Satz linearer Differentialgleichungen. Die vollständige Herleitung und Nomenklatur sind [redacted 1985] zu entnehmen. Im folgenden werden die für das particle tracking relevanten Grundlagen wiederholt und wo nötig erweitert.

2.1 Das Gesetz von Darcy

Aus der Impulserhaltung folgt mit den Annahmen, dass

1. äussere Kraftfelder durch Druckgradienten und die Erdbeschleunigung beschrieben werden,
2. Trägheitskräfte vernachlässigt werden und
3. der Strömungswiderstand proportional dem Volumenfluss q (= Darcy-Geschwindigkeit) ist,

das Darcy-Gesetz in der Form:

2-1 $$q^k = -K^{kl} \frac{\partial}{\partial x^l} \left(\frac{p}{\rho g} + x^3\right)$$

q^k	$[\frac{m^3}{s}/m^2]$	Darcy-Geschwindigkeit
K^{kl}	[m/s]	Permeabilitätstensor
p	[kg m/s²]	Druck
ρ	[kg/m³]	Dichte
g	[m/s²]	Erdbeschleunigung

Der Ausdruck $(\frac{p}{\rho g} + x^3)$ wird piezometrische Höhe genannt.

Er stellt ein Potential dar.

2-2 $h = (\frac{p}{\rho g} + x^3)$ Definition des Potentials

2.2 Die Kontinuitätsgleichung

Aus der Massenbilanz für stationäre inkompressible Strömungen erhält man:

2-3 $$\frac{\partial}{\partial x^k} \cdot q^k + Q = 0$$

Q [m³/s] Volumen - Quell/Senke-Term

Die Kombination von (2-1), (2-2) und (2-3) liefert

2-4 $$- \frac{\partial}{\partial x^k} \left(K^{kl} \frac{\partial h}{\partial x^l} \right) + Q = 0 \ ,$$

und mittels Definition der Porosität erhält man die Beziehung zwischen Darcy-Fluss und Fliessgeschwindigkeit.

2-5 $v^k = \frac{1}{\epsilon} q^k$ Fliessgeschwindigkeit

ϵ [1] Porosität des Mediums

2.3 Stromlinien

Im stationären Fall sind Trajektorien entlang Fliesswegen identisch den Stromlinien.

Stromlinien sind allgemein definiert durch das Tangentialfeld an das Geschwindigkeitsfeld zu einem gegebenen Zeitpunkt.

2-6 $\frac{\partial s^k(t,\lambda)}{\partial \lambda} = v^k(s^i,t)$ Definition der Stromlinienfunktion $s^k(t,\lambda)$

2-7 $\frac{dx^k(t)}{dt} = v^k(x^i,t)$ Definition des Fliessweges $x^k(t)$

Bei Stationarität ist die Geschwindigkeit $\vec{v}$ unabhängig von t; somit werden (2-6) und (2-7) identisch.

In einem 2-dimensionalen stationären Modell ist es möglich, die Bahn des Fliessweges als Linien konstanter Stromfunktion zu berechnen, indem man die Stromfunktion analog der Potentialfunktion bestimmt. Dieses Vorgehen scheitert meist daran, dass die Randbedingungen nicht einfach formulierbar sind.

Wir werden uns im folgenden an die zeitunabhängige Definition (2-8) halten.

2-8 $$\frac{dx^k}{dt} = v^k(x^i)$$

dies ergibt in Kombination mit (2-5) und (2-1)

2-9 $$\frac{dx^k}{dt} = - \frac{1}{\epsilon} K^{kl} \frac{\partial}{\partial x^l} h(x^i)$$

2.4 Lösung mittels FEM

Das Potentialproblem (2-4) wird mit der Methode der finiten Elemente gelöst. Die stationären Potentiale $h(x^i)$ sind der Ausgangspunkt zu Lösung von (2-9). Es liegt nahe, in (2-9) für $h(x^i)$ dieselben Interpolationsfunktionen anzusetzen wie für deren Berechnung.

[redacted] benützt im Code FEM301 quadratische isoparametrische Elemente; i.e. die Geometrie eines Elementes und die Interpolation des Potentials innerhalb dieses Elementes geschehen durch dieselben Ansatzfunktionen. Die Geometrie eines Elementes wird durch die Koordinatentransformation

2-10 $$x^k(s^i) = \sum_{\nu=1}^{n} N_\nu(s^i) \cdot x_\nu^k$$ approximiert.

Das Potential wird dargestellt durch

2-11 $$h(x^k) = \sum_{\nu=1}^{n} N_\nu(s^i(x^k))\, h_\nu$$

mit
$k = 1, 3$ Raumdimension des Modells
$i = 1, m$ Raumdimension des verwendeten Elementes
$\nu = 1, n$

x^k : globale Koordinaten

s^i : lokale Koordinaten

$N_\nu(s^i)$: Ansatzfunktionen in den Knoten ν

n : Anzahl Knoten

x_ν^k : Koordinaten des Knotens ν

h_ν : berechnetes Potential in den Knoten

Die Koordinatentransformation $x^k(s^i)$ ist im Normalfall eine Abbildung der gekrümmten Geometrie des Elementes auf ein Polyeder mit einfacher Geometrie. Es werden folgende Bedingungen an sie gestellt:

- $x^k(s^i)$ ist injektiv:

$$x^k(s_1^i) = x_1^k \;,\; x^k(s_2^i) = x_1^k \;\Longrightarrow\; s_1^i = s_2^i$$

- falls $m = 3$ ist, i.e. das Element ist echt 3d: $s^i(x^k)$ existiert und ist injektiv.

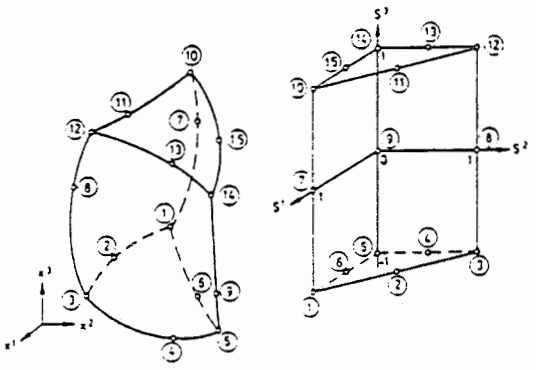


Fig. 1: Koordinatentransformation
Abbildung des gekrümmten Elementes auf ein reguläres Polyeder

2.4.1 Das Geschwindigkeitsfeld

Die Verwendung isoparametrischer Elemente führt auch für das Geschwindigkeitsfeld, Gleichung (2-9), zu einer Darstellung analog zur Potentialdarstellung und der Koordinatentransformation. Dazu werden noch einige differentialgeometrische Definitionen benötigt. Sie sollen im folgenden kurz erläutert werden.

Funktionalmatrix

Die Ableitung der Koordinatentransformation nach den lokalen Koordinaten ist definiert durch:

2-12 $$\frac{\partial x^k}{\partial s^i} = D_i^k$$ Funktional- oder Jacobimatrix

für i=1,3 wird die Funktionaldeterminante bestimmt durch

2-13 $$| D_i^k | = \det \left(\frac{\partial x^k}{\partial s^i} \right)$$

Metrischer Tensor

Um Umkehrabbildungen von "singulären" Elementen (≙ 2d- und 1d-Elemente) berechnen zu können, wird der metrische Tensor benötigt.

2-14 $$g_{im} = \frac{\partial x^k}{\partial s^i} \frac{\partial x^l}{\partial s^m} \delta_{kl}$$ 2-fach kovarianter metrischer Tensor

und

2-15 $$g^{kn} \delta_n^i g_{il} = \delta_l^k$$, mit g^{kn} = 2-fach kontravarianter Tensor

wobei δ_{kl} das Kroneckersymbol darstellt.

Mit Hilfe dieser Mittel lassen sich einige wichtige Grössen herleiten. Wir werden die im folgenden benötigten aufführen und versuchen, sie plausibel herzuleiten.

Inverse der Funktionalmatrix

Bei 3d-Elementen gewinnt man die Inverse Funktionalmatrix ganz einfach durch Matrix-Inversion.

2-16 $$\frac{\partial s^i}{\partial x^k} = \left(\frac{\partial x^k}{\partial s^i} \right)^{-1} = (D^{-1})^i_k \quad ,$$

da D^k_i eine invertierbare 3x3 Matrix ist.

Bei singulären Elementen erhält man die Inverse über den inversen metrischen Tensor.

2-17 $$\frac{\partial s^i}{\partial x^k} = g^{im}\, \delta_{kl}\, \frac{\partial x^l}{\partial s^m} \quad \left(\hat{=} (D^{-1})^i_k \right)$$

Beweis

$$\delta^p_k = \frac{\partial x^p}{\partial x^k} = \frac{\partial x^p}{\partial s^q}\frac{\partial s^q}{\partial x^k} \qquad | \cdot \frac{\partial x^l}{\partial s^m}\delta_{pl}$$

$$\underbrace{\delta^p_k\ \delta_{pl}}_{\delta_{kl}}\frac{\partial x^l}{\partial s^m} = \underbrace{\frac{\partial x^p}{\partial s^q}\frac{\partial x^l}{\partial s^m}\delta_{pl}}_{g_{qm}}\frac{\partial s^q}{\partial x^k} \qquad | \cdot g^{im}$$

$$g^{im}\,\delta_{kl}\frac{\partial x^l}{\partial s^m} = \underbrace{g^{im}\,g_{qm}}_{\delta^i_q}\frac{\partial s^q}{\partial x^k}$$

$$\delta^i_q\ \frac{\partial s^q}{\partial x^k} = \frac{\partial s^i}{\partial x^k}$$

Transformation von kovarianten Vektoren

Ein kovarianter Vektor ist durch sein Transformationsverhalten definiert. Wir gehen den umgekehrten Weg und fragen uns, wie ein in globalen Koordinaten kovarianter Vektor in lokalen Koordinaten aussieht. Als typisches Beispiel eines kovarianten Vektors gilt der Gradientoperator (z.B. der Potentialgradient).

Vor: $\frac{\partial}{\partial x^k}$ ---?---→ $\frac{\partial}{\partial s^i}$ ist gesucht

2-18 $$\frac{\partial}{\partial s^i} = \frac{\partial}{\partial x^k}\frac{\partial x^k}{\partial s^i}$$

2-19 $$\frac{\partial}{\partial x^k} = \frac{\partial s^i}{\partial x^k}\frac{\partial}{\partial s^i}$$

Beispiele

2-20 $$\frac{\partial h}{\partial x^k} = (D^{-1})^i_k \frac{\partial h}{\partial s^i}$$ Potentialgradient

2-21 Flächennormale n_k zu $F(x^k) = 0$

$$n_k = \frac{\partial F}{\partial x^k} = (D^{-1})^i_k \frac{\partial F}{\partial s^i}$$

Transformation von kontravarianten Vektoren

Analog zu obigen Transformationen leiten sich die Transformationen kontravarianter Vektoren her. Beispiel für diesen Fall ist der Geschwindigkeitsvektor.

Vor: ∂x^k ---?---→ ∂s^i ist gesucht

2-22 $$\partial x^k = \frac{\partial x^k}{\partial s^i}\partial s^i$$

2-23 $$\partial s^i = \frac{\partial s^i}{\partial x^k}\partial x^k$$

Beispiele

2-24 $$v^k = \frac{\partial x^k}{\partial t} = \frac{\partial x^k}{\partial s^i}\frac{\partial s^i}{\partial t}$$ Geschwindigkeitsvektor

2-25 Tangentiale τ^k an Raumkurve $x^k(r)$

$$\tau^k = \frac{\partial x^k}{\partial r} = \frac{\partial x^k}{\partial s^i}\frac{\partial s^i}{\partial r}$$

mit $s^i(r)$ der Parametrisierung der Kurve.

Wir nehmen noch einige Definitionen vorweg und erhalten für Gleichung (2-9) eine explizite Darstellung; i.e. dass die Differentialgleichung nur noch von einem Koordinatentripel abhängt.

2-26 $$u^i = \frac{\partial s^i}{\partial t}$$ lokale Geschwindigkeit

2-27 $$j_k = \frac{\partial h}{\partial x^k}$$ globaler Potentialgradient

2-28 $$l_i = \frac{\partial h}{\partial s^i}$$ lokaler Potentialgradient

2-29 $$\frac{\partial s^i}{\partial x^k} = \begin{cases} \left(\frac{\partial x^k}{\partial s^i}\right)^{-1} & , \; i,k = 1,3 \\ g^{im}\, \delta_{kl}\, \frac{\partial x^l}{\partial s^m} & , \; \mathrm{Max}(i,m)<3 \end{cases} = (D^{-1})^i_k$$

somit für (2-9)

2-30 $$\frac{\partial s^i}{\partial t} = -\frac{1}{\epsilon}\, \frac{\partial s^i}{\partial x^k}\, K^{kl}\, \frac{\partial s^m}{\partial x^l}\, \frac{\partial h}{\partial s^m}$$

Die Vorteile dieser Darstellung sind offensichtlich:

- Explizität der Funktionen $h(s^i)$ und $x^k(s^i)$; die Differentialgleichung ist von der Form

  $$\frac{\partial s^i}{\partial t} = F^i(s^m).$$

  Sie lässt sich mit gängigen Methoden wie Euler-Verfahren oder Runge-Kutta-Integration lösen.

- $F^i(s^m_k)$ ist elementweise analytisch. Ein Element ist im s^k-Koordinatenraum ein Polyeder; die Grenzen sind exakt berechenbar und der Uebergang zum nächsten Element ist einfach zu handhaben.

- Die lineare Interpolation der Funktion $s^k(t)$ entspricht in globalen Koordinaten einer Interpolation mittels Ansatzfunktionen. Eine gekrümmte Elementgeometrie schlägt sich auch in der Bahn des Fliessweges nieder. (Kann auch nachteilig sein!)

- Es ist möglich mit adaptiver Schrittweite zu integrieren, so dass der jeweiligen Feinheit des Elements Rechnung getragen wird. Die Anzahl Integrationsschritte ist einfach abzuschätzen; dies ist numerisch von grossem Vorteil.

Wir werden im nächsten Kapitel, wo die numerische Lösung besprochen wird, einen anschaulichen Einblick gewinnen. Des weiteren soll noch ein Approximationsverfahren erläutert werden, von dem ebenfalls in der Numerik Gebrauch gemacht wird.

2.5 Newton-Raphson-Algorithmus

Die Beziehung (2-10) $x^k = x^k(s^i)$ ist, wie sie im Code FEM301 benützt wird, quadratisch in allen Komponenten. Die Gleichung ist gewöhnlich nicht analytisch invertierbar. Ihre Gutartigkeit (injektiv) erlaubt uns, in den

meisten Fällen ein einfaches und schnell konvergentes Verfahren zur Lösung anzuwenden.

Basierend auf dem Newton-Verfahren bei R^1-R^1 Abbildungen (siehe Figur 2) lässt sich für Abbildungen $R^m \rightarrow R^n$, mit $m \leq n$, folgende Methode herleiten:

Vor: $x^k = x^k(s^i)$ $\quad k = 1,n \quad i = 1,m \quad m \leq n$

Gesucht: s^i_o, so dass $x^k(s^i_o) = x^k_o$

Algorithmus:

$$\Delta x^k_\nu = x^k_o - x^k_\nu$$

$$\Delta s^i_\nu = s^i_{\nu+1} - s^i_\nu \qquad \text{und}$$

$$\frac{\Delta x^k_\mu}{\Delta s^i_\mu} \simeq \frac{\partial x^k}{\partial s^i}(s^l_\mu) \qquad \text{damit folgt}$$

2-31 $$s^i_{\nu+1} = s^i_\nu + \frac{\partial s^i}{\partial x^k}(s^l_\nu) \cdot (x^k_o - x^k_\nu)$$

Zur Berechnung von $\frac{\partial x^i}{\partial x^k}$ werden die im obigen Abschnitt hergeleiteten Beziehungen benötigt.

Voraussetzung für Konvergenz des Verfahrens ist eine strikt positive (oder negative) Funktionaldeterminante.

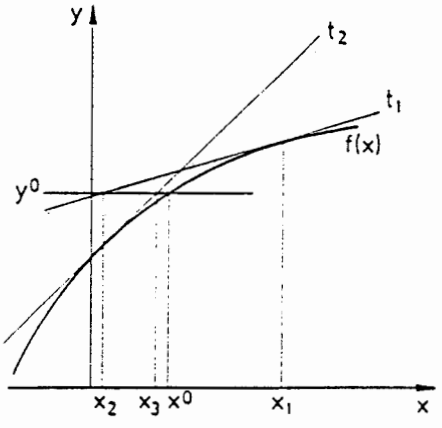


Fig. 2: Newton-Verfahren:
Folge x_1, x_2 ... konvergiert nach x^o, mit $y^o = f(x^o)$. x_{n+1} ist gegeben durch den Schnittpunkt der Tangente t_n an $f(x_n)$ mit der Geraden $y = y^o$.

3. NUMERISCHES LOESUNGSVORGEHEN

Bevor vorgängig beschriebene Theorie in numerische Verfahren umgesetzt wird, soll ein Katalog aufgestellt werden, in dem steht, was bereits bekannt ist, welche Anforderungen gestellt sind, welche Resultate gefragt sind. Anhand dieses Kataloges werden Schritt für Schritt numerische Methoden bereitgestellt und diskutiert.

3.1 Anforderungskatalog

Ausgangspunkt ist ein 3-dimensionaler gesättigter Grundwasserträger. Bekannt sind

- die topologischen Daten (Ausdehnung, Einteilung in finite Elemente, Stollen etc.)
- die hydrogeologischen Daten (Randbedingungen, Porositäten, Permeabilitäten, Brunnen, Zuflüsse)
- die numerisch berechneten Potentiale (FEM301) in den Knoten.

Gegeben sind weiter

- Startpunkte von Trajektorien an interessierenden Stellen (potentielle Endlager, Kontaminationsschwerpunkte).

Gesucht sind

- die Fliesswege, beginnend bei den Startpunkten (particle tracks)
- die Fliesszeiten
- evt. lokale Fliessgeschwindigkeiten
- evt. Gründe für Berechnungsabbruch

3.2 Lösungen

Jede Trajektorie wird nach folgendem Konzept berechnet:

i) Zum Startpunkt werden zugehöriges Element und lokale Koordinaten bestimmt.

ii) Lösung der Differentialgleichung (2-30) für die lokale Fliessgeschwindigkeit, bis die Trajektorie den Elementrand erreicht.

iii) Bestimmen des nächsten Elementes und den zugehörigen lokalen Koordinaten und weiterfahren nach ii), bis ein Abbruchkriterium wirksam wird.

3.2.1 Lokalisieren des Startpunktes

Das Startelement wird aus der Klasse von Elementen bestimmt, die den Knoten enthalten, der dem Startpunkt am nächsten ist. Für diese Elemente werden die zum Startpunkt gehörenden lokalen Koordinaten nach der Newton-Raphson-Methode berechnet, bis ein Element gefunden wird, dessen Koordinaten innerhalb des Elementes liegen.

3.2.2 Punkt im Polyeder-Test

Die Berechnung, ob ein Punkt innerhalb oder ausserhalb eines Elementes liegt, kann bei krummliniger Geometrie nur approximativ geschehen. Der Umweg über die Berechnung der lokalen Koordinaten liefert eine einfache Lösung, da in lokalen Koordinaten alle Elemente Polyeder darstellen.

Man betrachtet das Polyeder als Schnitt von Halbräumen und braucht nur zu entscheiden, ob die Koordinate im richtigen Halbraum liegt. Jeder Halbraum wird definiert durch die Ebene, die eine Polyeder-Fläche enthält, und man ordnet beispielsweise der das Polyeder enthaltenden Hälfte ein positives Vorzeichen zu.

3.2.3 Integration der Differentialgleichung

Die Gleichung (2-30) vom Typ $\frac{\partial s^i}{\partial t} = F^i(s^k)$ wird mittels Euler-Verfahren integriert, wobei wahlweise eine Runge-Kutta-Approximation eingeschaltet werden kann.

Euler-Verfahren:

3-1 $$s^i_{\nu+1} = s^i_\nu + F^i(s^k_\nu) \cdot \Delta t$$

Runge-Kutta-Verfahren:

Die Grösse $F^i(s^k_\nu)$ wird so gemittelt, dass die Fehler bis zur 3. Ordnung verschwinden.

$F^i(s^k_\nu)$ ---→ $\hat{F}^i(s^k_\nu)$ mit

3-2 $\hat{F}^i(s^k_\nu) = \frac{1}{6}(f^i_1 + 4\,f^i_2 + f^i_3)$ und

$f^i_1 = F^i(s^k_\nu)$

$f^i_2 = F^i(s^k_\nu + \frac{1}{2}\Delta t f^k_1)$

$f^i_3 = F^i(s^k_\nu + 2\Delta t f^k_2 - \Delta t f^k_1)$

Der Zeitschritt Δt wird so gewählt, dass bei jedem Integrationsschritt dieselbe räumliche Schrittweite resultiert. Dies hat den Vorteil, dass die Anzahl Schritte abschätzbar ist und bei extremen Geschwindigkeiten (z.B. in der Nähe von Brunnen oder Senken) keine zu weiten (oder zu geringen) Sprünge entstehen.

3-3 $| F^i(s^k_\nu) | \cdot \Delta t = \Delta s = \text{konstant}$

Nach jedem Integrationsschritt wird ermittelt, ob der neue Punkt noch innerhalb des Elementes liegt (Punkt im Polyeder-Test). Punkte ausserhalb werden entlang des letzten Schrittes auf den Elementrand zurückgesetzt.

3.2.4 Uebergang zum nächsten Element

Wenn die Trajektorie den Elementrand erreicht, muss ein neues, folgendes Element zur Fortsetzung bestimmt werden. Dabei ist Typ des Randes, ob es eine Fläche, Kante oder ein Knoten ist, von Bedeutung. Bei Kante oder Knoten kommen mehrere Elemente in Frage. Zur Behandlung dieses Falles wird auf den folgenden Abschnitt verwiesen. Ueblicherweise ist der Rand 2-dimensional. Um das angrenzende Element zu bestimmen, wird vorgängig die sogenannte Connection Matrix berechnet. Sie enthält zu jedem Element die an die Seitenfläche grenzenden Nachbarelemente.

Beispiel

Element Nr. 17 habe an Fläche Nr. 3 Element Nr. 13 als Nachbar mit Fläche Nr. 2:

ICON(3, 17) = 13
ICON(2, 13) = 17

Das relativ zeitaufwendige Erstellen der Connection Matrix zahlt sich vor allem dann aus, wenn mehrere Berechnungen am selben Modell (i.e. ungeänderte Topologie) gemacht werden, oder viele Trajektorien während eines Durchlaufes zu berechnen sind. Zu diesem Zweck ist es vorteilhaft, sie als File abzuspeichern.

Ist das folgende Element bestimmt, werden die lokalen Eintrittskoordinaten bestimmt und die Integration fortgesetzt. Es ist möglich, die neuen Koordinaten direkt

aus den alten zu berechnen. Dazu wird neben Austritts- und Eintrittsfläche die Orientierung der beiden Flächen zueinander benötigt. Weitere Schwierigkeiten ergeben sich bei den sogenannten "pinched elements", wo eine Dreiecksfläche aus einer "zusammengedrückten" Viereckfläche gebildet wird. Deshalb wird auf dieses Vorgehen verzichtet, und die Koordinaten werden nach der Newton-Raphson-Methode berechnet.

3.2.5 1- und 2-dimensionale Elemente

Elemente niederer Dimension bedürfen einer Spezialbehandlung. Sie sind mathematisch singulär, i.e. sie entstehen als Grenzfall eines unendlich dünnen 3-dimensionalen oder 2-dimensionalen Elementes. Es werden einige Möglichkeiten des Vorgehens diskutiert:

1. Fall: Ignorieren

Die einfachste Lösung dieses Problems ist, nichts zu tun. Dass man das darf, erklärt sich dadurch:

- Es entstehen keine quantitativen Lücken im Modell; alle Potentiale werden berücksichtigt.

Der Nachteil ist, dass der Einfluss hochkonduktiver Elemente unberücksichtigt bleibt. Für eine Näherung ist dieses Vorgehen brauchbar.

2. Fall: Analog wie 3-dimensionale Elemente

Als zweite Möglichkeit bietet sich an, die 2- und 1-dimensionalen Elemente ungeachtet ihrer Dimension zu behandeln. Damit wird der im 1. Fall vernachlässigte Effekt berücksichtigt; andererseits wird wegen der fehlenden Geschwindigkeitskomponente "quer" zum Element die Trajektorie das Element nur am Elementrand verlassen können.

3. Fall: Kombination von Fall 1 und Fall 2

Da beide obigen Vorgehen nicht ganz befriedigen, wird ein Verfahren gewählt, wo der Einfluss aller Elemente berücksichtigt wird, die den aktuellen Punkt enthalten. Typischerweise liegen 2-dimensionale Elemente zwischen 3-dimensionalen und 1-dimensionalen Elementen auf Kanten von 3-dimensionalen Elementen, so dass mindestens 3 Elemente einen Einfluss haben. Um sich für ein Element zu entscheiden, geht man davon aus, dass die Trajektorie den schnellsten Weg wählt. Sie führt ihren Weg in dem Element fort, das die Trajektorie aufnimmt und die grösste Geschwindigkeit aufweist. Erfolgt ein Schritt entlang des Randes, wird diese Auswahl nach jedem Integrationsschritt neu getroffen, da die Nachbarelemente

noch beteiligt sind. Nach diesem Vorgehen werden sowohl 1- und 2-dimensionale Elemente einbezogen wie auch die von den 3-dimensionalen Elementen stammenden "Querkomponenten" berücksichtigt.

Nach demselben Vorgehen wird ein Element ausgewählt, wo der Rand eine Kante oder ein Eckknoten ist.

3.2.6 Oszillationen

An Elementrändern treten noch weitere Schwierigkeiten auf. Wegen der Unstetigkeit des Geschwindigkeitsfeldes am Elementrand entstehen vornehmlich bei starken Kontrasten der Konduktivität entgegengesetzt gerichtete Geschwindigkeitsvektoren, so dass kein Element die Trajektorie "akzeptiert". Fährt man in diesem Fall mit der grössten Geschwindigkeit weiter, wird der nächste Integrationsschritt wieder zurückführen, so dass auf dem Rand eine Zickzack-Linie entsteht. In manchen Fällen findet die Trajektorie nicht mehr weg; sie oszilliert.

Der Grund für Oszillationen kann auch bedingt sein durch Potentialmulden, die auf ungenügend feine Diskretisierung zurückzuführen sind. In diesem Fall wird die Berechnung abgebrochen, was physikalisch sinnvoll ist. Ein Versuch, die Oszillation auf dem Rand zu glätten, soll noch erläutert werden.

In allen den betreffenden Rand enthaltenden Elementen wird der Geschwindigkeitsvektor untersucht. Falls der Vektor aus dem Element hinauszeigt, wird nur die Komponente entlang des Randes berücksichtigt. Es sind drei Typen von Rändern nach ihrer Dimension zu unterscheiden. Liegt der Punkt auf einer Elemntseitenfläche (2-dimensionaler Rand), erhält man die gesuchte Geschwindigkeitskomponente durch Projektion des Vektors auf diese Fläche. Liegt der Punkt auf einer Elementkante (1-dimensionaler Rand) oder in einem Eckknoten (0-dimensionaler Rand), wird versucht, den Vektor nach absteigender Dimension der beteiligten Ränder zu projizieren. Bei einer Kante sind Seitenflächen und bei einem Eckknoten Seitenflächen und Kanten miteinzubeziehen. Ein Vektor wird so oft projiziert, bis eine Komponente entlang des Randes ins Element zeigt. Auf diese Weise werden die Freiheitsgrade minimal eingeschränkt. Bei einem Eckknoten als Rand kann der Fall eintreten, dass keine Komponente ins Element zeigt. Dieses Element ist auszuschliessen.

3.2.7 Konzept zur Lösung von Problemen am Elementrand

Es liegt nahe, die niedrig dimensionalen Elemente (s. 3.2.5) und die Oszillationen (s. 3.2.6) nach demselben

Vorgehen zu behandeln, da beides Phänomene sind, die an Elementrändern auftreten. Dabei stellt sich heraus, dass zwei Konzepte (i-ii) verfolgt werden können. Ausgangspunkt für beide Konzepte ist:

- Der Punkt liegt auf einem Rand (2-, 1-, 0-dimensionaler Rand, nach (3.2.6))

- Fallunterscheidung bei niedrig dimensionalen Elementen
  a) es liegen keine 1- und 2-dimensionalen Elemente vor
  b) es liegen 1- und 2-dimensionale Elemente vor

- Die Geschwindigkeitsvektoren werden nach dem Vorgehen in (3.2.6) projiziert. Hernach werden Klassen von Geschwindigkeitsvektoren gebildet. Eine erste Klasse enthält alle Vektoren, eine zweite Klasse enthält die projizierten Vektoren, eine dritte Klasse enthält die nicht-projizierten Vektoren.

i) Prinzip des grössten Freiheitsgrades (Normalfall)
Das Konzept beruht darauf, dass ein nicht-projizierter Vektor, da er mehr Freiheitsgrade besitzt, einem projizierten Vektor bevorzugt wird. Es wird nach 2 Fällen unterschieden:

a) Keine 2-, 1-dimensionalen Elemente
Die dritte Klasse der nicht projizierten Vektoren hat Priorität vor der ersten Klasse.

b) 2-, 1-dimensionale Elemente vorhanden
Die erste Klasse (alle Vektoren) wird untersucht.

ii) Prinzip des schnellsten Weges (Konservativer Fall)
Ohne Fallunterscheidung wird die erste Klasse untersucht.

Ziel ist es, aus der gewählten Klasse den Geschwindigkeitsvektor mit maximalem Betrag auszuwählen und die Trajektorie im zugehörigen Element fortzusetzen. Falls der Integrationsschritt auf dem Rand verläuft, muss nach einem Schritt das ganze Vorgehen wiederholt werden.

3.2.8 Abbruch-Kriterien

Die Berechnung einer Trajektorie endet im Idealfall dann, wenn sie das Gebiet verlässt oder in eine vorgegebene Senke gelangt.

Das Verlassen des Gebietes ist numerisch und geometrisch wohl definiert.

Eine Senke kann lokalisiert werden, indem die Distanz nach mehreren Integrationsschritten berechnet wird, und diese Distanz eine vorgegebene Mindestreichweite betragen soll.

Eine weitere Möglichkeit ist, die Richtungsänderung zweier aufeinander folgender Geschwindigkeitsvektoren zu berechnen und nur eine maximale Aenderung zuzulassen.

Die maximale Anzahl Integrationsschritte kann begrenzt werden.

3.3 Formelzusammenstellung

Die folgende Formelsammlung soll den in (3.2) beschriebenen Lösungsvorgang kurz zusammenfassen, um den Uebergang zur Computer-Programmierung zu erleichtern. Gleichzeitig wird die Schreibweise in Matrixdarstellung angefügt.

	Tensor		Vektor, Matrix
3-4	v_k	$\dashrightarrow$	$v_k = \vec{v}$
3-5	A_i^k	$\dashrightarrow$	$a_{ik} = A$
3-6	$A_i^k \cdot v_k$	$\dashrightarrow$	$a_{ik}\, v_k = A\vec{v}$
3-7	$A_k^i \cdot v_k$	$\dashrightarrow$	$a_{ki}\, v_k = A^T \vec{v}$

Funktionalmatrix (F)

3-8 $\dfrac{\partial x^k}{\partial s^i} \dashrightarrow d_{ik} = D \; ; \quad F = D$

Inverse der Funktionalmatrix (F^{-1})

3-9a $\dfrac{\partial s^i}{\partial x^k} \dashrightarrow (d^{-1})_{ki} = D^{-1} \; ; \quad F^{-1} = D^{-1}$

3-9b $\dfrac{\partial s^i}{\partial x^k} = g^{im}\, \delta_{kl} \cdot \dfrac{\partial x^l}{\partial s^m} \dashrightarrow F^{-1} = (D \cdot D^T)^{-1} \cdot D$

Globaler Potentialgradient ($\vec{j}$)

3-10 $\dfrac{\partial h}{\partial x^k} = \dfrac{\partial h}{\partial s^i} \dfrac{\partial s^i}{\partial x^k} \dashrightarrow \vec{j} = F^{-1}\, \vec{I}$

mit $\vec{I}$: lokaler Potentialgradient

Lokale Geschwindigkeit $(\vec{u})$

3-11 $$u^i = \frac{\partial s^i}{\partial t} = \frac{\partial s^i}{\partial x^k} \cdot v^k \quad \dashrightarrow \quad (F^{-1})^T \vec{v}$$

mit $\vec{v}$: globale Geschwindigkeit

Projektion auf Fläche $S(\vec{x}) = 0$

Normalenvektor: $\vec{n}$

3-12 $$n_k = \frac{\partial S}{\partial x^k} = \frac{\partial s^i}{\partial x^k} \frac{\partial S}{\partial s^i} \quad \dashrightarrow \quad \vec{n} = (F^{-1}) \cdot \vec{m}$$

mit $m_i = \frac{\partial S}{\partial s^i}$: lokale Normale

3-13 $$\vec{v}_p = \vec{v} - \frac{(\vec{v}, \vec{n})}{||\vec{n}||^2} \cdot \vec{n}$$

mit $\vec{v}$: globaler Vektor

$\vec{v}_p$: Projektion

Projektion auf Kante $\vec{x}(r)$

Tangentialvektor: $\vec{\tau}$

3-14 $$\tau^k = \frac{\partial x^k}{\partial s^i} \frac{\partial s^i}{\partial r} \quad \dashrightarrow \quad \vec{\tau} = F^T \vec{\sigma}$$

mit $\sigma^i = \frac{\partial s^i}{\partial r}$: lokale Tangente

3-15 $$\vec{v}_p = \frac{(\vec{v}, \vec{\tau})}{||\vec{\tau}||^2} \vec{\tau}$$

Differentialgleichung der lokalen Trajektorie

3-16 $$\vec{s} = - \frac{1}{\epsilon} (F^{-1})^T K F^{-1} \cdot \vec{I}$$

"Lokaler Permeabilitätstensor" K_{lokal}

3-17 $$K_{lokal} = (F^{-1})^T K F^{-1}$$

4. PROGRAMMBESCHREIBUNG

Das Programm TRACK ist als Postprocessor zu FEM301 konzipiert. Es benützt genau dieselben Eingabedaten für die Elementtopologie und die Koordinaten. Die Formate sind in der Dokumentation zu FEM301 beschrieben. Die Parametereingabe wurde um die für das Tracking notwendige Porosität erweitert.

Ein zusätzliches Inputfile enthält die Startpunkte und verschiedene Programmoptionen, ein weiteres die von FEM301 berechneten Potentiale.

Die Output-Files sind ständiger Aenderung unterworfen. Es werden 3 Output-Files generiert:

- ein File enthält die in (2.) beschriebene Connection-Matrix,
- ein zweites File liefert Informationen über den Berechnungsablauf,
- ein drittes enthält die Trajektorienkoordinaten, die zum Zeichnen der Trajektorien weiterverwendet werden.

4.1 Input-Files

Vier Files werden von FEM301 übernommen.

- Elementdaten, unverändert
- Koordinaten, unverändert
- Potentiale, unverändert
- Parameter (Randbedingungen, Permeabilitäten, Porositäten)

Wie erwähnt, wird im Parameterfile zusätzlich die Porosität eingegeben.

Das neue Format ist:

```
.
.
.
PERMEABILITIES AND POROSITIES

NP   POR   K1   K2   K3   K4   K5   K6
.
.
```

mit
NP: Klassennumerierung
POR: Porosität
K1..K6: Permeabilitätstensor

Eingabe bei 2- und 1-dimensionalen Elementen

Bei ein- und zweidimensionalen Elementen können Aquifermächtigkeit oder Querschnittsfläche nicht eingegeben werden. Damit die Fliessgeschwindigkeit richtig berechnet wird, sind sie in der Porosität zu berücksichtigen.

Im zweidimensionalen Element mit der Mächtigkeit a [m] ist die fiktive Porosität ϵ_2

4-1 $$\epsilon_2 = \frac{\epsilon}{a} \quad ,$$

im eindimensionalen Element mit der Querschnittsfläche F [m²]

4-2 $$\epsilon_1 = \frac{\epsilon}{F} \quad .$$

- Startpunkte

  Die für den Benützer relevanten Optionen werden besprochen.

  START
  NOTRACK MAXSTEP
  NUMB XIN YIN ZIN
  .
  .
  .

  wobei
  NOTRACK: Anzahl Startpunkte
  MAXSTEP: Begrenzung der Anzahl Integrationsschritte
  NUMB: Numerierung der Trajektorie
  XIN,YIN,ZIN: Koordinaten der Startpunkte

  Beispiel: Startpunkte

```
STEP
  0 1

DISCRETE
  0 01

OUTPUT DISTANCE
  0 01

BOUND
  0

KUTTA OFF
  1

START
     4   500
     1   -10.05000000     -6.00000000
     2   -10.06223587     -5 92274575
     3   -10.09774575     -5.85305369
     4   -10.15305369     -5.79774575
```

4.2 Output-Files

Es werden 3 Ausgabe-Files generiert. Die Bedeutung kann den Beispielen entnommen werden.

- Connection Matrix, dient als Zwischenspeicher für spätere Berechnung; ist für den Benützer nicht von Bedeutung
- Elementinformationen, verschiedene nützliche Angaben über den Verlauf der Trajektorien innerhalb der Elemente
- Koordinaten, Trajektorienkoordinaten in einfacher Darstellung zur Verwendung durch Plotprogramme.

Beispiel: Elementinformationen

```
.....................................................
* PARTICLE TRACKING FOR FINITE ELEMENTS              *
* IN THREE DIMENSIONS                                *
* VERSION          2.1                               *
* ETH/VAX-MCI      09-SEP-1988                       *
* TIME: 29-SEP-88  14:06      USER: U8064            *
.....................................................

   Input files:   ETHZ_A:U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_ELM
                  ETHZ_A:U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_XYZ
                  ETHZ_A:U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_PAR
                  ETHZ_A:U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_RES
                  ETHZ_A:U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_STA
                  ETHZ_A:U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_CON

   Output files:  ETHZ_A:U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_INF
                  ETHZ_A:U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_PAT

$ READ INPUT FILES
$ TIME USED FOR THIS STEP:    16.28 SECONDS
$ TOTAL TIME USED:            16.28 SECONDS


    RESULTS OF THE TRACKING
    .......................


  NUMBER OF TRAJECTORIES.. . . . . .  :  4
  MAXIMUM NUMBER OF STEPS PER TRAJECTORY  :  500


  LOCAL STEP SIZE . . . . . .  :  .1000
  SUBDISCRETE REGION DISTANCE. . . . . .  :  .0100 [m]
  MINIMUM DISTANCE OF TRAJ. COORDINATES.  :  .0150 [m]
  BOUNDARY THRESHOLD SELECTION  :  0
    (0 = ENTER FLUX TAKING 3D-ELEMENTS,
     1 = FOLLOW HIGHEST BOUNDARY VELOCITY)
  RUNGE KUTTA INTEGRATION . . . . . .  :  1
    (0 = OFF,
     1 = ON )


     TRACK NO.    1

ELEMENT K-CLASS NITTOT   LENGTH [m]    TIME [y]     VELOCITY [m/y]  CUM LEN [m]    CUM TIME [y]   MEAN VEL [m]
    575       2     14   9.23130E-02  8.09066E-07   1.14098E+05   9.23130E-02   8.09066E-07   1.14098E+05   (** 2D TRACK **)
    215       2     22   8.00534E-02  7.92051E-07   1.01071E+05   1.72366E-01   1.60112E-06   1.07654E+05   (** 2D TRACK **)
   1169       2     35   6.37407E-02  7.66178E-07   8.31930E+04   2.36107E-01   2.36730E-06   9.97371E+04   (** 2D TRACK **)
   1170       2     48   2.22240E-01  2.89512E-06   7.67638E+04   4.58347E-01   5.26241E-06   8.70983E+04   (** 2D TRACK **)
   1173       2     60   4.01986E-01  6.27992E-06   6.40113E+04   8.60333E-01   1.15423E-05   7.45372E+04   (** 2D TRACK **)
   1172       2     75   6.81390E-01  1.19725E-05   5.69129E+04   1.54172E+00   2.35148E-05   6.55638E+04   (** 2D TRACK **)
   1174       2     81   1.79213E-01  3.18016E-06   5.63536E+04   1.72094E+00   2.66950E-05   6.44686E+04   (** 2D TRACK **)
   1144       2    102   7.32147E-01  1.29679E-05   5.64586E+04   2.45308E+00   3.96629E-05   6.18484E+04   (** 2D TRACK **)
   1141       7    118   5.86795E-01  9.43172E-06   6.22150E+04   3.03988E+00   4.90946E-05   6.19188E+04   (** 2D TRACK **)
   1139       7    127   2.53425E-01  3.88015E-06   6.53139E+04   3.29331E+00   5.29747E-05   6.21675E+04   (** 2D TRACK **)
    206       7    139   4.10592E-01  5.78521E-06   7.09727E+04   3.70390E+00   5.87599E-05   6.30344E+04   (** 2D TRACK **)
    205       7    154   3.51942E-01  4.49673E-06   7.82663E+04   4.05584E+00   6.32567E-05   6.41172E+04   (** 2D TRACK **)
   1138       7    157   2.83969E-02  3.42627E-07   8.28800E+04   4.08424E+00   6.35993E-05   6.42183E+04   (** 2D TRACK **)
   1138       7    159   1.81184E-02  1.77480E-07   1.02087E+05   4.10236E+00   6.37768E-05   6.43237E+04   (** 2D TRACK **)
     94       7    162   9.23522E-03  1.21105E-07   7.62579E+04   4.11159E+00   6.38979E-05   6.43463E+04   (** 2D TRACK **)
    814       7    179   1.28277E-01  1.38277E-06   9.27680E+04   4.23987E+00   6.52806E-05   6.49483E+04   (** 2D TRACK **)
    815       7    191   9.42509E-02  9.22767E-07   1.02150E+05   4.33413E+00   6.62034E-05   6.54669E+04   (** 2D TRACK **)
     95       7    196   4.97821E-02  2.53497E-07   1.96381E+05   4.38391E+00   6.64569E-05   6.59662E+04   (** 2D TRACK **)

              TOTAL :                                             4.38391E+00   6.64569E-05   6.59662E+04

 TRACK ENDS IN CORNER NODE OR LOCAL SINK

$      TRACK NO.    1 CALCULATED
$ TIME USED FOR THIS STEP:        97 SECONDS
$ TOTAL TIME USED.              17.25 SECONDS
```

Beispiel: Trajektorienkoordinaten

```
File name:    :ETHZ_A.U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_PAT

Date:         09-SEP-88  14:06
User:         U8064
Program:      TRACK - 2.0
Input files:  :ETHZ_A.U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_ELM
              :ETHZ_A.U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_XYZ
              :ETHZ_A.U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_PAR
              :ETHZ_A.U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_RES
              :ETHZ_A.U8064.FORTRAN.FEM.TRACK.NAMES.NEW.T1_STA

PATH      1
    1       -10.250          -6.000          0.000    0
   50        -9.567          -6.105          0.000    0
   51        -9.531          -6.115          0.000    0
   52        -9.495          -6.126          0.000    0
   53        -9.458          -6.137          0.000    0
   54        -9.422          -6.148          0.000    0
   55        -9.386          -6.160          0.000    0
   56        -9.350          -6.171          0.000    0
   57        -9.313          -6.182          0.000    0
   58        -9.277          -6.194          0.000    0
   59        -9.240          -6.205          0.000    0
   60        -9.218          -6.212          0.000    0
   62        -9.170          -6.230          0.000    0
   63        -9.121          -6.248          0.000    0
```

4.3 Reduktion der Modelldimension

Es ist möglich, auch 2- oder 1-dimensionale Modelle zu rechnen. Das Programm entscheidet anhand der Input-Daten aus dem Element-File, von welcher Dimension das Modell ist.

Bei reduzierter Dimensionalität übernehmen die 2-dimensionalen Elemente die Rolle der 3-dimensionalen Elemente und bedürfen keiner Behandlung als Randelemente mehr. Es sind sowohl flache (i.e. nur x,y-Koordinaten) wie auch "krumme" (x,y,z-Koordinaten), im 3-dimensionalen eingebettete Modelle, zugelassen. Die Eingabe der Daten ändert sich dabei nicht.

5. VERIFIKATION DES PROGRAMMES

Das Programm wird an zwei analytischen Beispielen getestet. Gewählt wird ein heterogener 2-dimensionaler Fall, da Heterogenität in der Regel mehr Schwierigkeiten bereitet als 3-Dimensionalität und keine 3-dimensionalen heterogenen analytischen Beispiele zum Vergleich verfügbar sind.

Abschliessend werden Ergebnisse zu einem operationellen Modell dargestellt. Ein Vergleich der Resultate mit den Resultaten eines andern Computercodes soll das Auftreten und die unterschiedliche Behandlung von Oszillationen verdeutlichen.

5.1 Analytischer Vergleichsfall

Das folgende Beispiel wurde für HYDROCOIN als Programmverifikation von [redacted] vorgeschlagen. Es handelt sich um den Zylinder im unendlich ausgedehnten Grundwasserleiter mit konstanter Grundströmung. Es werden die beiden Fälle eines hoch- und eines schwach-durchlässigen Zylinders betrachtet. Anhand zweier verschieden diskretisierter Netze wird die analytische Lösung mit 4 numerischen Berechnungsmethoden verglichen:

- Fall 1: analytische Geschwindigkeiten
- Fall 2: numerische Geschwindigkeiten
- Fall 3: analytische Potentiale
- Fall 4: numerische Potentiale

Als Vergleichspunkte werden an 2 Positionen die aufsummierten Fliesszeiten und die Fehler in den Positionen untersucht.

5.1.1 Problemstellung

Der Testfall basiert auf der analytischen Lösung für eine stationäre, inkompressible, zweidimensionale Strömung um ein kreisförmiges Gebiet mit unterschiedlicher Durchlässigkeit. Er beinhaltet sowohl Uebergänge zwischen Gebieten mit verschiedenen Aquifereigenschaften wie auch gekrümmte Trajektorien. Die geometrische Konfiguration ist den nachfolgenden Figuren zu entnehmen.

Die Materialeigenschaften seien isotrop.

5.1.2 Mathematisches Modell

Es gelten das Gesetz von Darcy

5-1 $$q^k = -K^{kl} \frac{\partial h}{\partial x^l} \quad ,$$

die Kontinuitätsgleichung

5-2 $$\frac{\partial}{\partial x^k} q^k = 0 \quad ,$$

wobei

$$K^{kl} = \delta^{kl} K_i \quad , \text{ für } \quad r < a$$

$$K^{kl} = \delta^{kl} K_a \quad , \text{ für } \quad r > a \quad ,$$

und die Trajektoriengleichung

5-3 $$v^k = \frac{1}{\epsilon} q^k = \frac{dx^k}{dt} \quad .$$

Auf der Grenzschicht $|\vec{x}| = a$ sei der Potentialübergang stetig.

5.1.3 Analytische Lösung

Für Potential (h), Geschwindigkeit (v^k) und Trajektorien (x,y) ergeben sich die analytischen Ausdrücke:

5-4 $$h = \begin{cases} -x + \dfrac{(K_i - K_a)}{(K_i + K_a)} \dfrac{a^2}{r^2} x & r > a \\ -x \dfrac{2K_a}{K_i + K_a} & r < a \end{cases}$$

5-5 $$v^1 = \begin{cases} \dfrac{K_a}{\epsilon} \left(1 - \dfrac{(K_i - K_a)}{(K_i + K_a)} a^2 \dfrac{y^2 - x^2}{r^4}\right. & r > a \\ \dfrac{K_i}{\epsilon} \dfrac{2K_a}{(K_i + K_a)} & r < a \end{cases}$$

$$v^2 = \begin{cases} \dfrac{K_a}{\epsilon} 2 \dfrac{(K_i - K_a)}{(K_i + K_a)} a^2 \dfrac{xy}{r^4} & r > a \\ 0 & r < a \end{cases}$$

5-6 $$y + \frac{a^2}{r^2} \frac{y}{} \frac{(K_i - K_a)}{(K_i + K_a)} = y_o = \text{konstant} \qquad r > a$$

$$y = y_o \frac{(K_i - K_a)}{2 K_i} \qquad r < a$$

Die Berechnung der Aufenthaltszeiten erfolgt numerisch aus (5-5) und (5-6).

5-7 $$t - t_o = \int_{x_o}^{x} \frac{dx'}{v^1}$$

5.1.4 Input-Parameter

a) Durchlässiger Zylinder

Neben der geometrischen Konfiguration sind noch folgende Materialeigenschaften anzugeben:

$K_i = 10^{-7}$ m/s
$K_a = 10^{-9}$ m/s
$\epsilon = 0.1$

Als Startpunkte werden gewählt:

No	x-Position [m]	y-Position [m]
1	-50.	10.
2	-50.	12.
3	-50.	14.
4	-50.	16.
5	-50.	18.
6	-50.	20.

b) Undurchlässiger Zylinder

Die Materialeigenschaften und Startpunkte sind wie folgt gewählt:

$K_i = 10^{-11}$ m/s
$K_a = 10^{-9}$ m/s

No	x-Position [m]	y-Position [m]
1	-50.	0.1
2	-50.	0.12
3	-50.	0.14
4	-50.	0.16
5	-50.	0.18
6	-50.	0.20
7	-50.	0.22
8	-50.	0.24
9	-50.	0.5
10	-50.	1.0

5.1.5 Output

Wie einleitend erwähnt, werden die analytischen Ergebnisse mit 4 numerischen Lösungen verglichen. In den vier Fällen werden an 2 Positionen die Fehler der numerischen Lösung untersucht. Es werden an den Positionen x = 0 m und x = 50 m je die kumulierten Zeiten und die y-Positionen berechnet. Die Ergebnisse werden tabellarisch zusammengefasst. Die Tabelle beinhaltet für jede Trajektorie:

- Tra.No : Trajektoriennummer
- X Start
  Y Start : Startposition
- Y (x = 0): Y-Position bei x = 0 m
- Err (Y0) : Fehler Y (x = 0) - $Y_{analytisch}$
- X Final
  Y Final : Endposition
- T (x = 0): kumulative Zeit bei x = 0 m
- Err (T0) : Fehler T(x = 0) - $T_{analytisch}$
- T Final : kumulative Zeit bei x = 50 m
- Err (TF) : Fehler T Final - $T_{analytisch}$

FE-Diskretisierung

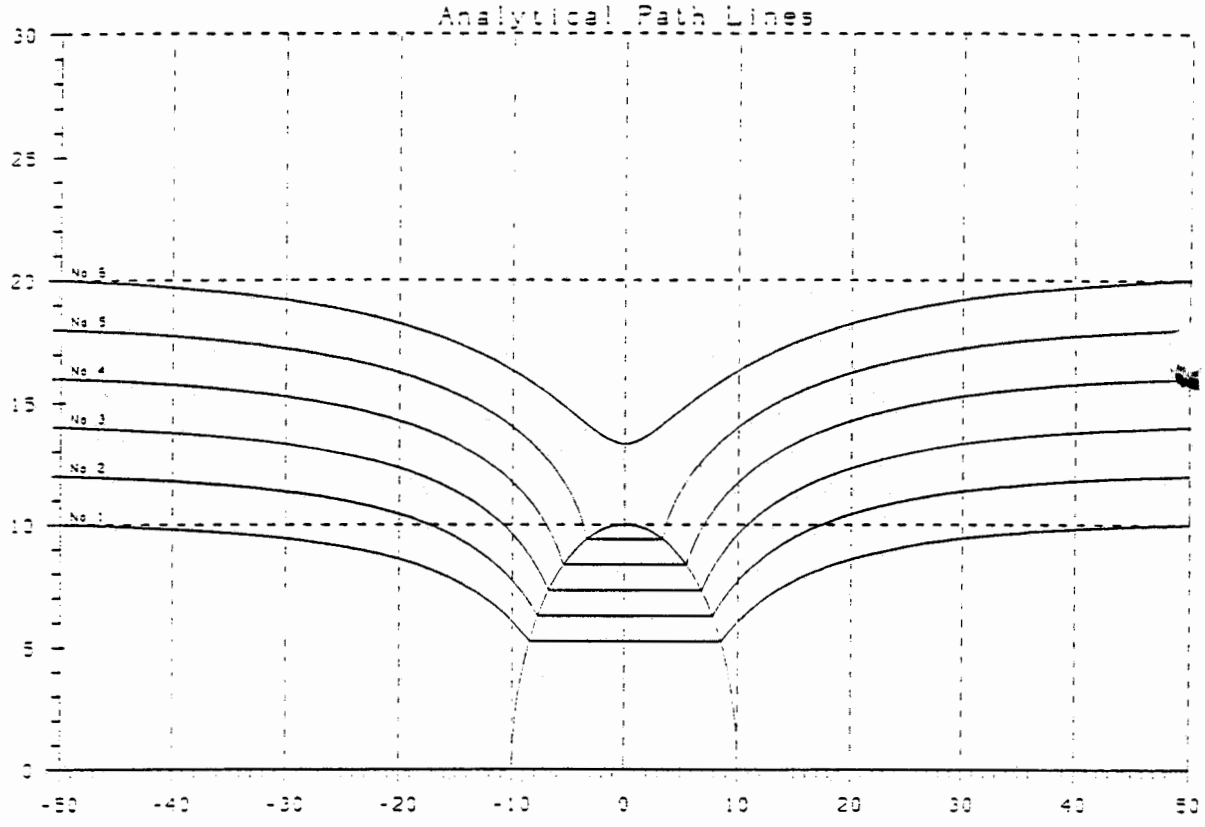
Das Problem wird anhand von zwei verschieden diskretisierten Netzen gerechnet. Die Elementunterteilung ist gestrichelt gezeichnet.

- Das erste Netz besteht aus 30 Elementen. Der Halbzylinder ist in 4 Elemente unterteilt.

- Das zweite Netz wurde feiner gewählt. Es besteht aus 226 Elementen und der Halbzylinder enthält 36 Elemente. Aus ersichtlichen Gründen (siehe unten) wurde die Uebergangszone sehr fein diskretisiert.

5.1.6 Durchlässiger Zylinder

i) Analytische Resultate



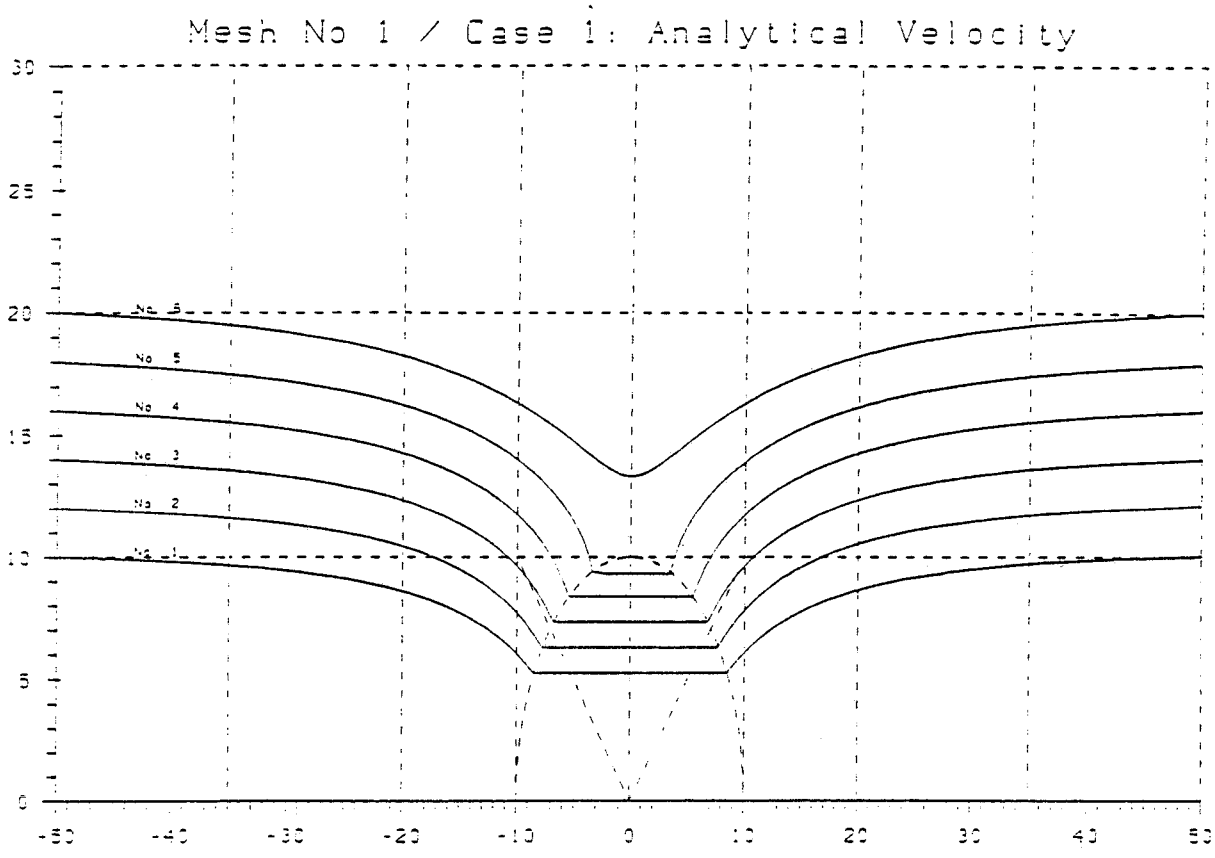
ANALYTICAL RESULTS

Units : Length [m] Time 1E09 [s]

Trg No	Y (X=0.)	Y (X=50.)	Time (X=0.)	Time (X=50.)
1	5.24038462	10.00000000	4.16621590	8.33243180
2	6.28465961	12.00000000	4.28973663	8.57947327
3	7.32704748	14.00000000	4.45747188	8.91494377
4	8.36737300	16.00000000	4.69921783	9.39843567
5	9.40550992	18.00000000	5.12931807	10.25863615
6	13.31363839	20.00000000	5.53240565	11.06481130

Fig. 3: analytische Trajektorien

ii) Grobes Netz

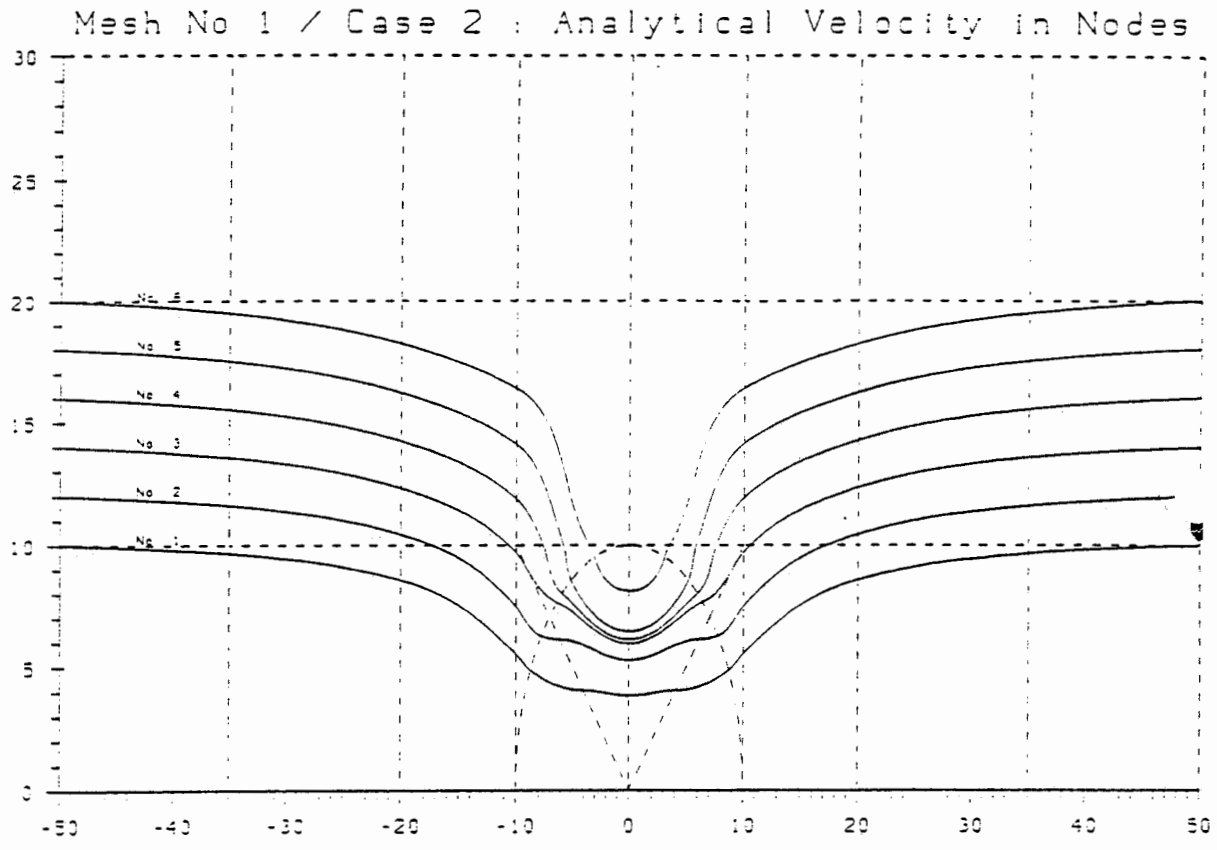


PARTICLE TRACKING SUMMARY (Mesh 1/Case 1 .Analytical Velocity)

Units : Length 1E+01 [m] Time 1E+10 [s]

Tra.No	1	2	3	4	5	6
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	1.0000001	1.2000001	1.4000001	1.6000001	1.8000001	2.0000001
Y (X=0)	.5261191	.6290551	.7341281	.8378961	.9329161	1.3316501
Err (Y0)	.0020811	.0005891	.0014231	.0011591	-.0076351	.0002861
X Final)	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final)	1.0060641	1.2114961	1.4022991	1.6003291	1.7925241	2.0000401
Err (YF)	.0060641	.0114961	.0022991	.0003291	-.0074761	.0000401
T (X=0)	.4166551	.4290041	.4456171	.4697051	.5138591	.5532551
Err (T0)	.0000331	.0000301	-.0001301	-.0002171	.0009281	.0000141
T Final	.8335261	.8587651	.8915081	.9395681	1.0244441	1.1064781
Err (TF)	.0002831	.0008171	.0000131	-.0002751	-.0014201	-.0000051

Fig. 4a: analytisches Geschwindigkeitsfeld

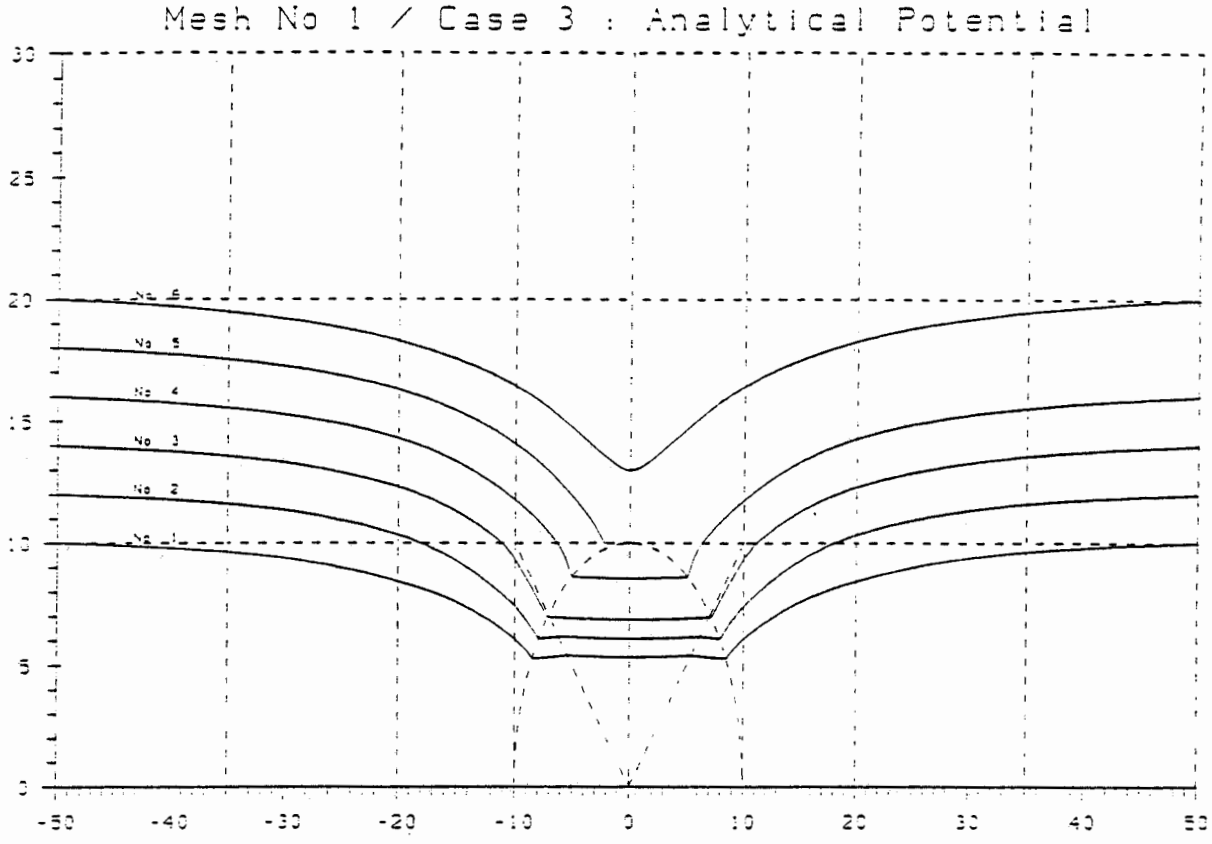


PARTICLE TRACKING SUMMARY (Mesh 1/Case 2 .Analytical Velocity in Nodes)

Units : Length 1E+01 [m] Time 1E+10 [s]

Trc.No	1	2	3	4	5	6
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	1.0000001	1.2000001	1.4000001	1.6000001	1.8000001	2.0000001
Y (X=0)	.3844421	.5298861	.5983531	.6149701	.6469331	.8122741
Err (Y0)	-.1395961	-.0985801	-.1343521	-.2217681	-.2936181	-.5190901
X Final)	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final)	1.0006261	1.2007691	1.4026811	1.6034811	1.8021291	2.0006011
Err (YF)	.0006261	.0007691	.0026811	.0034811	.0021291	.0006011
T (X=0)	.4194751	.4371801	.4668031	.5023861	.5448711	.5869621
Err (T0)	.0028531	.0082061	.0210561	.0324641	.0319401	.0337211
T Final	.8388621	.8743011	.9339781	1.0054031	1.0905131	1.1740101
Err (TF)	.0056191	.0163541	.0424831	.0655591	.0645501	.0675281

Fig. 4b: diskretisiertes Geschwindigkeitsfeld

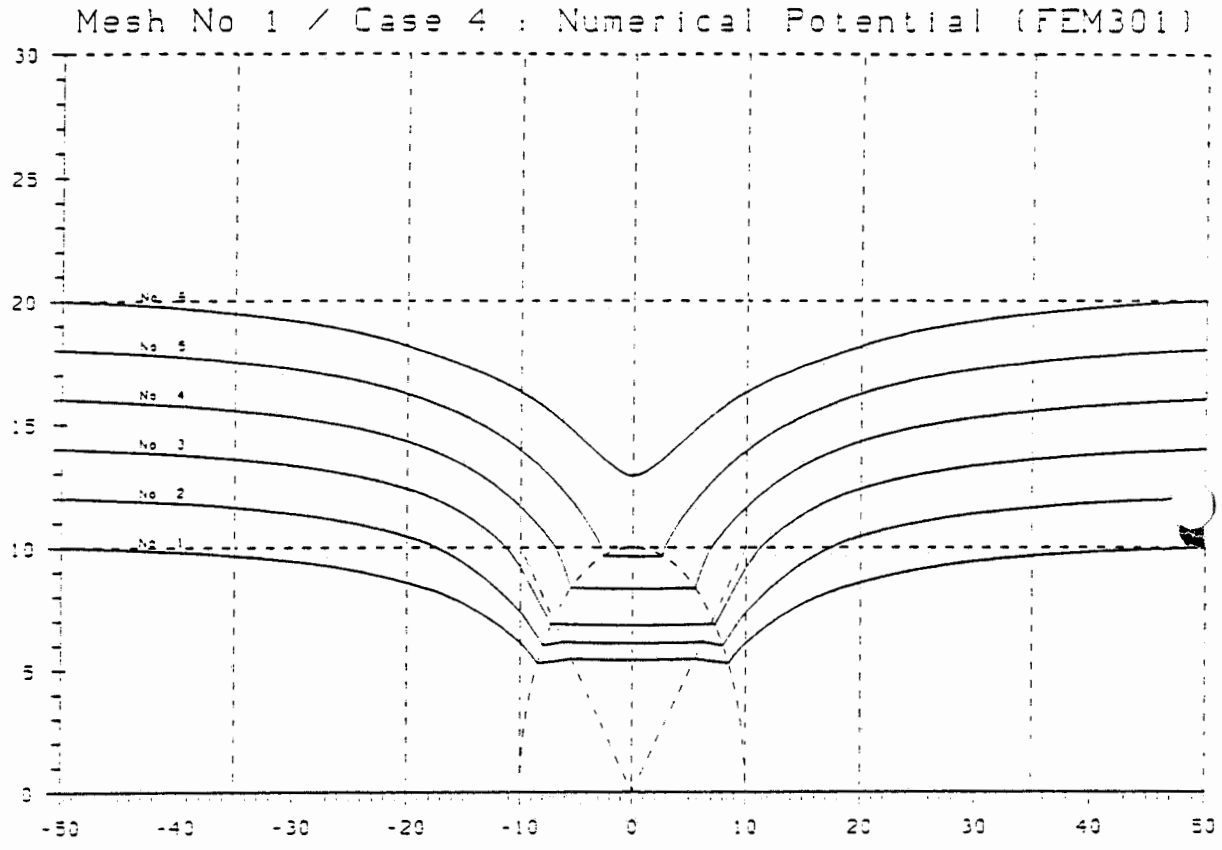


PARTICLE TRACKING SUMMARY (Mesh 1/Case 3 :Analytical Potential)

Units : Length 1E+01 [m] Time 1E+10 [s]

Trg.No	1	2	3	4	5	6
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	1.0000001	1.2000001	1.4000001	1.6000001	1.8000001	2.0000001
Y (X=0)	.5344711	.6091661	.6878801	.8567531	1.8000001	1.2993831
Err (Y0)	.0104321	-.0193001	-.0450251	.0200161	.0000001	-.0319811
X Final)	5.0000001	5.0000001	5.0000001	5.0000001	- 2265651	5.0000001
Y Final)	1.0003921	1.2006631	1.4002651	1.5998531	1.0000001	1.9998131
Err (YF)	.0003921	.0006631	.0002651	-.0001471	.0594491	-.0001871
T (X=0)	.4149071	.4270981	.4451961	.4734211	0.0000001	.5776871
Err (T0)	-.0017151	-.0018781	-.0005511	.0034991	0.0000001	.0244471
T Final	.8298131	.8542021	.8903961	.9468781	.5255511	1.1573281
Err (TF)	-.0034311	-.0037451	-.0010981	.0070351	.0250611	.0508461

Fig. 4c: diskretisiertes, analytisches Potentialfeld



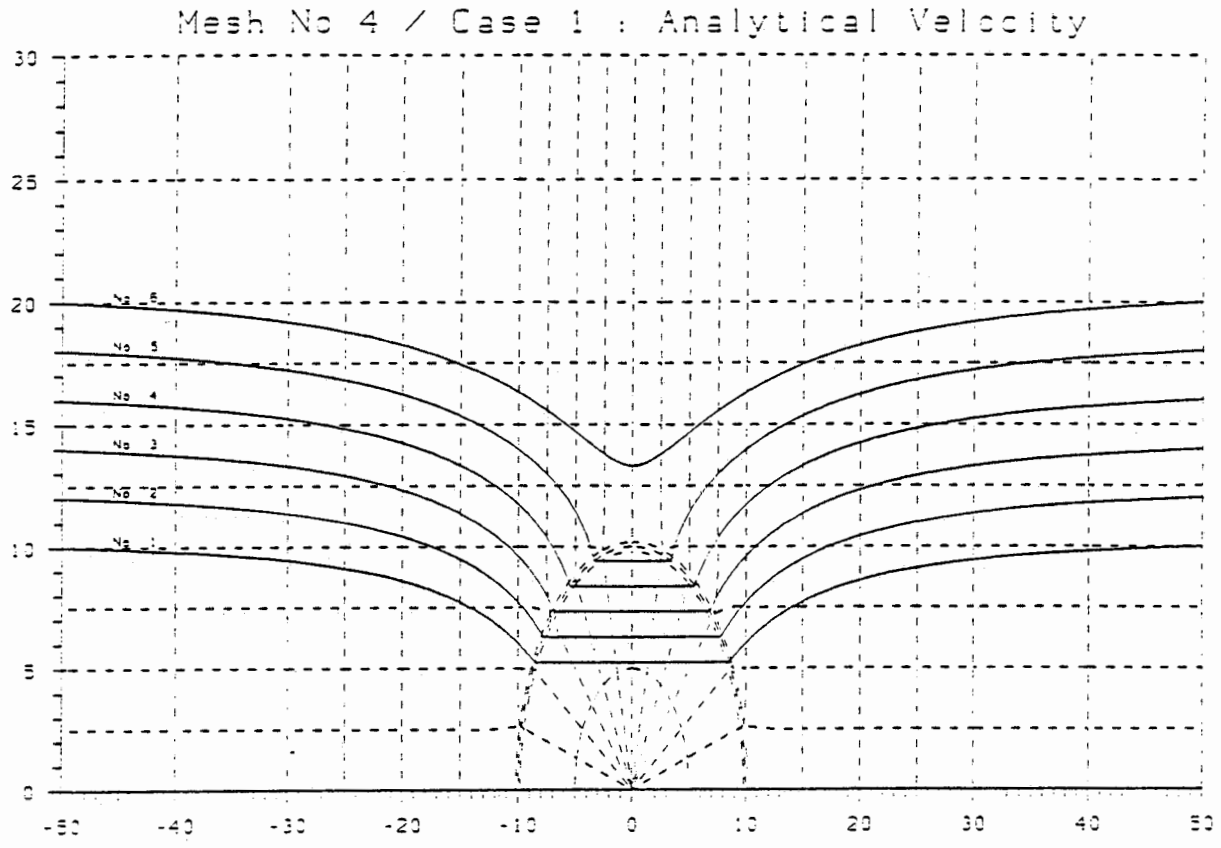
PARTICLE TRACKING SUMMARY (Mesh 1/Case 4 :Numerical Potential)

Units : Length 1E+01 [m] Time 1E+10 [s]

Tra.No	1	2	3	4	5	6
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	1.0000001	1.2000001	1.4000001	1.6000001	1.8000001	2.0000001
Y (X=0)	.5427611	.6099621	.6820941	.8310861	.9635191	1.2894421
Err (Y0)	.0187221	-.0185041	-.0506111	-.0056511	.0229681	-.0419221
X Final)	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final)	1.0002481	1.2006061	1.4000801	1.5998361	1.7999941	1.9999791
Err (Yf)	.0002481	.0006061	.0000801	-.0001641	-.0000061	-.0000211
T (X=0)	.4151981	.4274111	.4452501	.4701311	.5266041	.5699111
Err (T0)	-.0014241	-.0015631	-.0004971	.0002091	.0136721	.0166711
T Final	.8303761	.8548461	.8904771	.9402961	1.0533281	1.1407301
Err (Tf)	-.0028671	-.0031011	-.0010181	.0004521	.0274651	.0342491

Fig. 4d: numerisch berechnete Potentiale

iii) Feines Netz

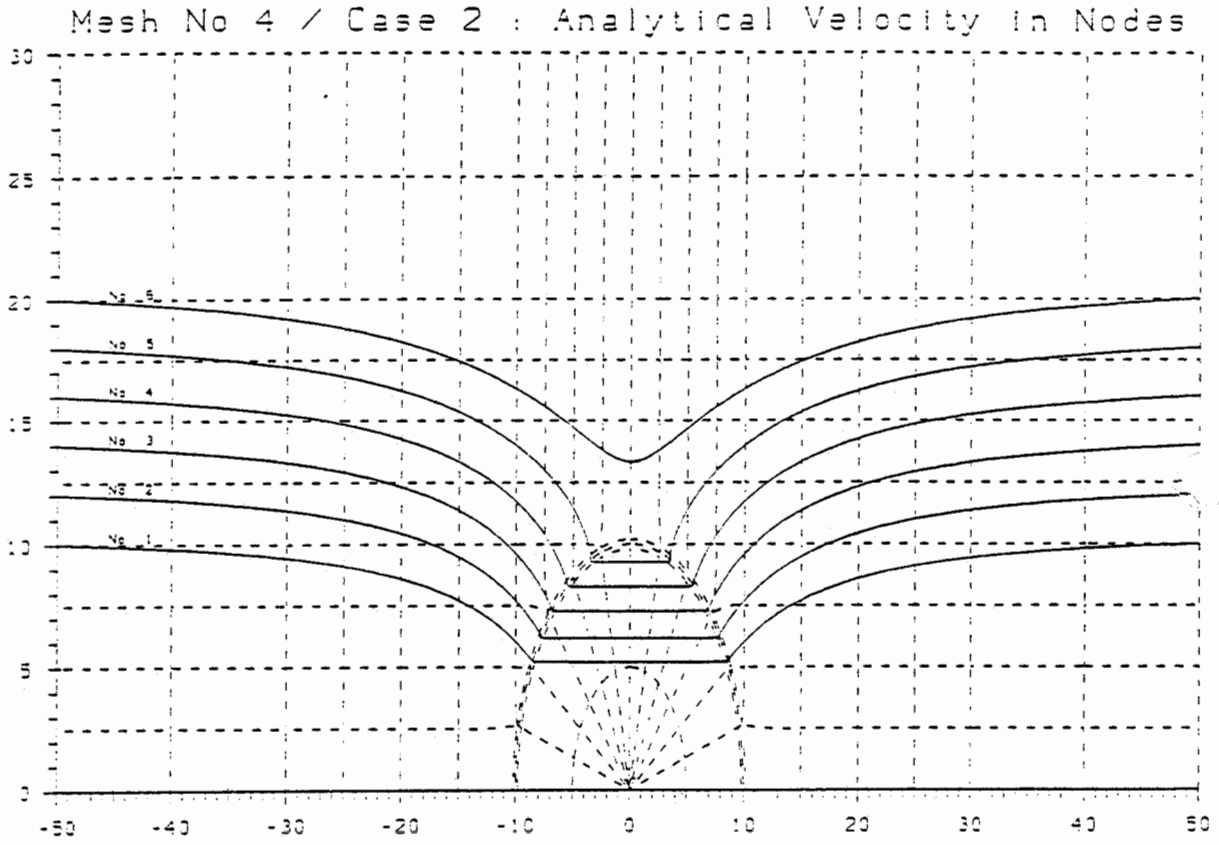


PARTICLE TRACKING SUMMARY (Mesh 4/Case 1 :Analytical Velocity)

Units : Length 1E+01 [m] Time 1E+10 [s]

Trg.No	1	2	3	4	5	6
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	1.0000001	1.2000001	1.4000001	1.6000001	1.8000001	2.0000001
Y (X=0)	.5241471	.6283921	.7325621	.8365861	.9403901	1.3313151
Err (Y0)	.0001081	-.0000741	-.0001431	-.0001511	-.0001611	-.0000481
X Final	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final	1.0000921	1.2001381	1.3998101	1.5998271	1.8003071	1.9999531
Err (YF)	.0000921	.0001381	-.0001901	-.0001731	.0003071	-.0000471
T (X=0)	.4166361	.4289941	.4457541	.4699341	5129261	.5532461
Err (T0)	.0000141	.0000201	.0000071	.0000121	-.0000061	.0000061
T Final	.8332341	.8579691	.8914651	.9398301	1.0260151	1.1064941
Err (TF)	-.0000091	.0000211	-.0000301	-.0000141	.0001511	.0000131

Fig. 5a: analytisches Geschwindigkeitsfeld

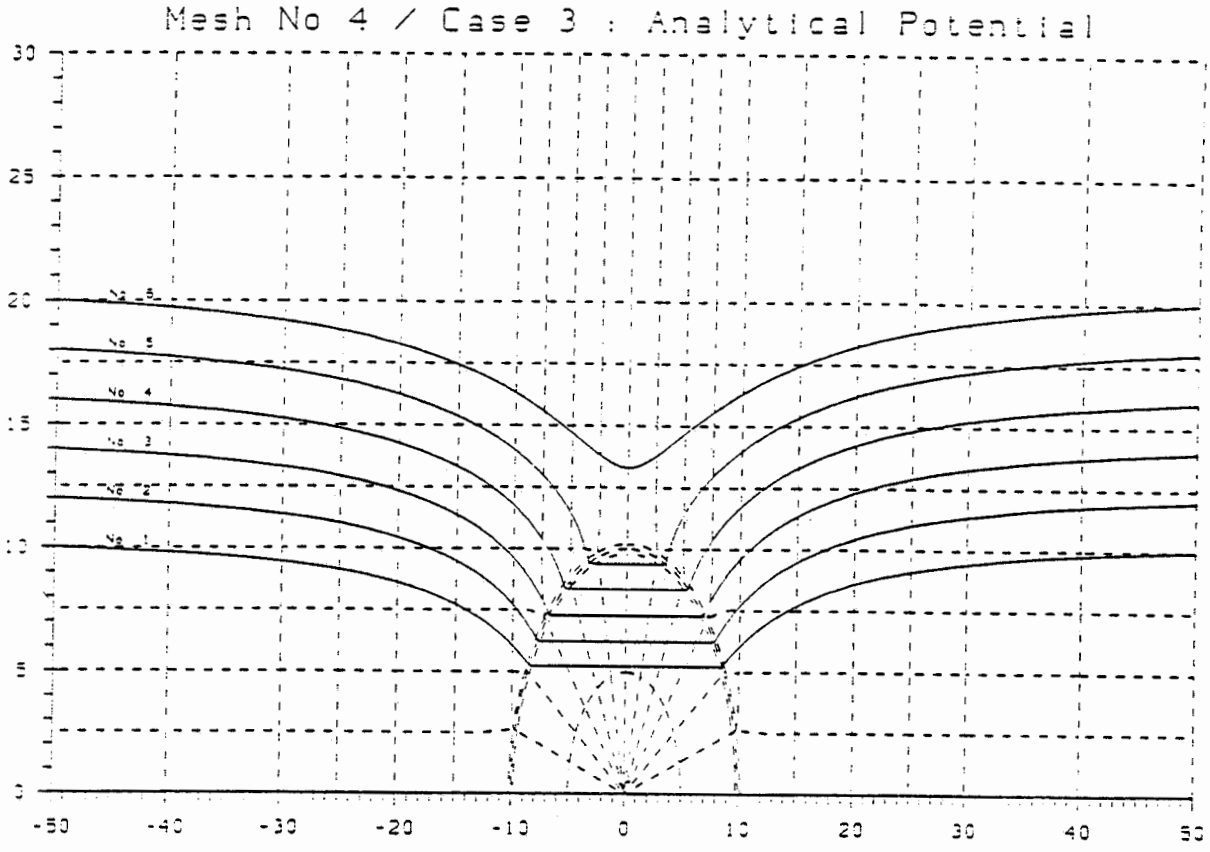


PARTICLE TRACKING SUMMARY (Mesh 4/Case 2 :Analytical Velocity in Nodes)

Units : Length 1E+01 [m] Time 1E+10 [s]

Tra.No	1	2	3	4	5	6
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	1.0000001	1.2000001	1.4000001	1.6000001	1.8000001	2.0000001
Y (X=0)	.5210471	.6191391	.7283381	.8279721	.9269621	1.3311301
Err (Y0)	-.0029911	-.0093271	-.0043671	-.0087661	-.0135891	-.0002341
X Final)	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final)	1.0001001	1.2000551	1.3999471	1.5999501	1.7999361	1.9999541
Err (YF)	.0001001	.0000551	-.0000531	-.0000501	-.0000641	-.0000461
T (X=0)	.4167681	.4293671	.4481991	.4705771	.5145491	.5533131
Err (T0)	.0001461	.0003931	.0004521	.0006551	.0018171	.0000731
T Final	.8335011	.8587031	.8923781	.9411431	1.0290941	1.1066281
Err (TF)	.0002581	.0007561	.0008831	.0012991	.0032311	.0001471

Fig. 5b: diskretisiertes Geschwindigkeitsfeld

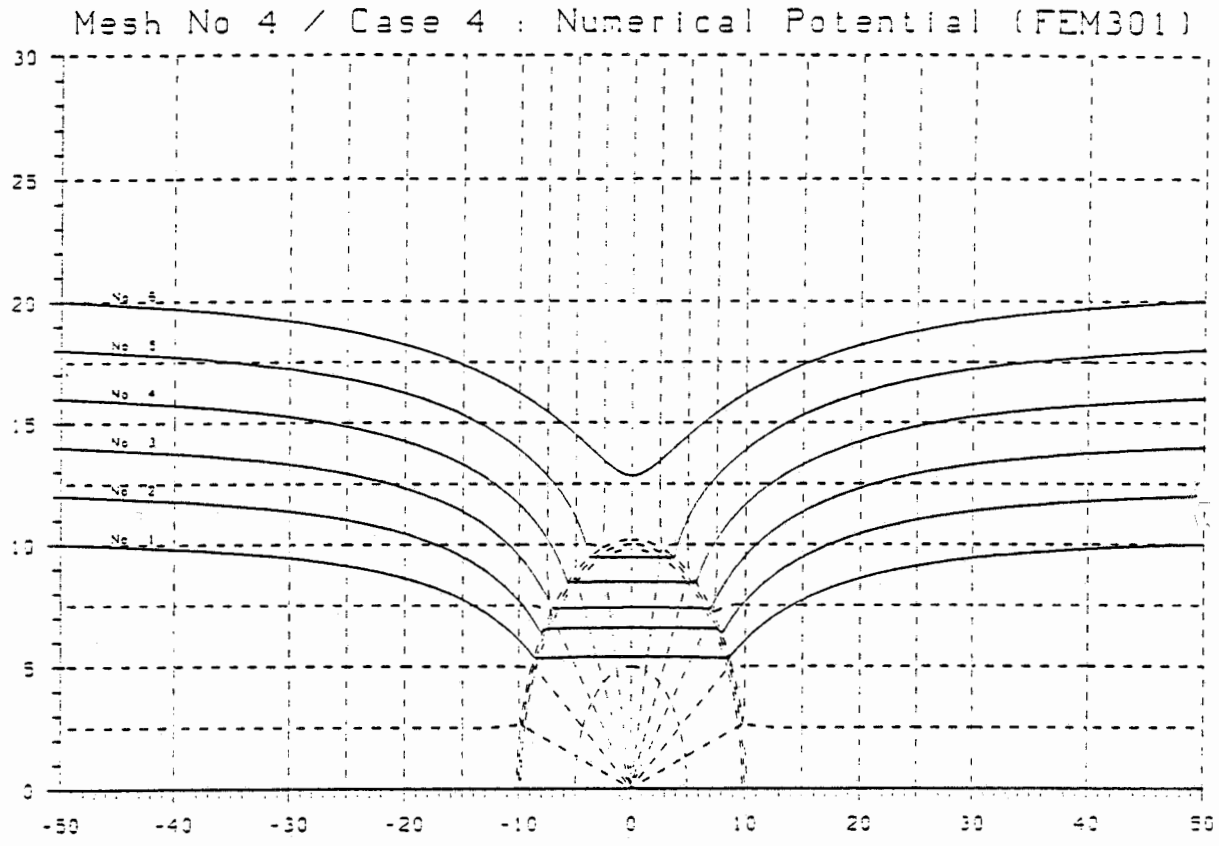


PARTICLE TRACKING SUMMARY (Mesh 4/Case 3 :Analytical Potential)

Units : Length 1E+01 [m] Time 1E+10 [s]

Tra.No	1	2	3	4	5	6
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	1.0000001	1.2000001	1.4000001	1.6000001	1.8000001	2.0000001
Y (X=0)	.5249851	.6232951	.7295201	.8364261	.9386881	1.3308041
Err (YO)	.0009471	-.0051711	-.0031851	-.0003111	-.0018631	-.0005591
X Final)	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final)	1.0000931	1.2001431	1.3998921	1.5998911	1.7999491	1.9999551
Err (YF)	.0000931	.0001431	-.0001081	-.0001091	-.0000511	-.0000451
T (X=0)	.4153311	.4275631	.4442051	.4681061	.5098781	.5533281
Err (TO)	-.0012911	-.0014111	-.0015421	-.0018151	-.0030541	.0000871
T Final	.8306281	.8551081	.8883801	.9361921	1.0197781	1.1066681
Err (TF)	-.0026151	-.0028391	-.0031141	-.0036521	-.0060861	.0001871

Fig. 5c: diskretisiertes, analytisches Potentialfeld



PARTICLE TRACKING SUMMARY (Mesh 4/Case 4 :Numerical Potential)

Units : Length 1E+01 [m] Time 1E+10 [s]

Tra.No	1	2	3	4	5	6
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	1.0000001	1.2000001	1.4000001	1.6000001	1.8000001	2.0000001
Y (X=0)	.5386841	.6589981	.7415741	.8479461	.9475811	1.2832631
Err (Y0)	.0146461	.0305321	.0088691	.0112091	.0070301	-.0481011
X Final)	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final)	1.0000641	1.2001831	1.3998571	1.6000161	1.7999081	1.9999261
Err (YF)	.0000641	.0001831	-.0001431	.0000161	-.0000921	-.0000741
T (X=0)	.4128661	.4262911	.4416801	.4653191	.5042391	.5620021
Err (T0)	-.0037561	-.0026831	-.0040671	-.0046031	-.0086931	.0087611
T Final	.8257041	.8525891	.8833241	.9306331	1.0084711	1.1240401
Err (TF)	-.0075391	-.0053581	-.0081701	-.0092101	-.0173931	.0175591

Fig. 5d: numerisch berechnete Potentiale

5.1.7 Diskussion der Resultate (durchlässiger Zylinder)

i) Fall 1: Analytisches Geschwindigkeitsfeld

Bei Vorgabe der analytischen Geschwindigkeit in jedem Punkt wird der Integrationsalgorithmus getestet. Dies beinhaltet Koordinatentransformation, lokaler Integrationsschritt und Rücktransformation sowie Bestimmen des Elementrandpunktes bei Erreichen der Elementgrenze.

Beide Netze ergeben fast exakte Ergebnisse. Der Fehler liegt in der Grössenordnung 10^{-4}. Die Unterschiede, die sich zwischen den beiden Netzen ergeben, sind auf die adaptive Schrittweite zurückzuführen. Im feinen Netz wird mit kleineren Schritten integriert.

ii) Fall 2: Diskretisiertes, analytisches Geschwindigkeitsfeld

Die Geschwindigkeit wird in jedem Punkt aus den Knotenwerten interpoliert.

Die Unterschiede zwischen den beiden Netzen sind frappant. Das grobe Netz liefert für die Uebergangszone völlig falsche Ergebnisse. Der Grund dafür ist, dass einerseits das Geschwindigkeitsfeld in diesem Bereich unstetig ist und andererseits die Interpolationsfunktionen in diesem Bereich den Geschwindigkeitsverlauf schlecht darstellen. Durch feines Diskretisieren dieser Zone im zweiten Netz werden diese Fehler stark reduziert.

iii) Fall 3: Analytisches Potentialfeld

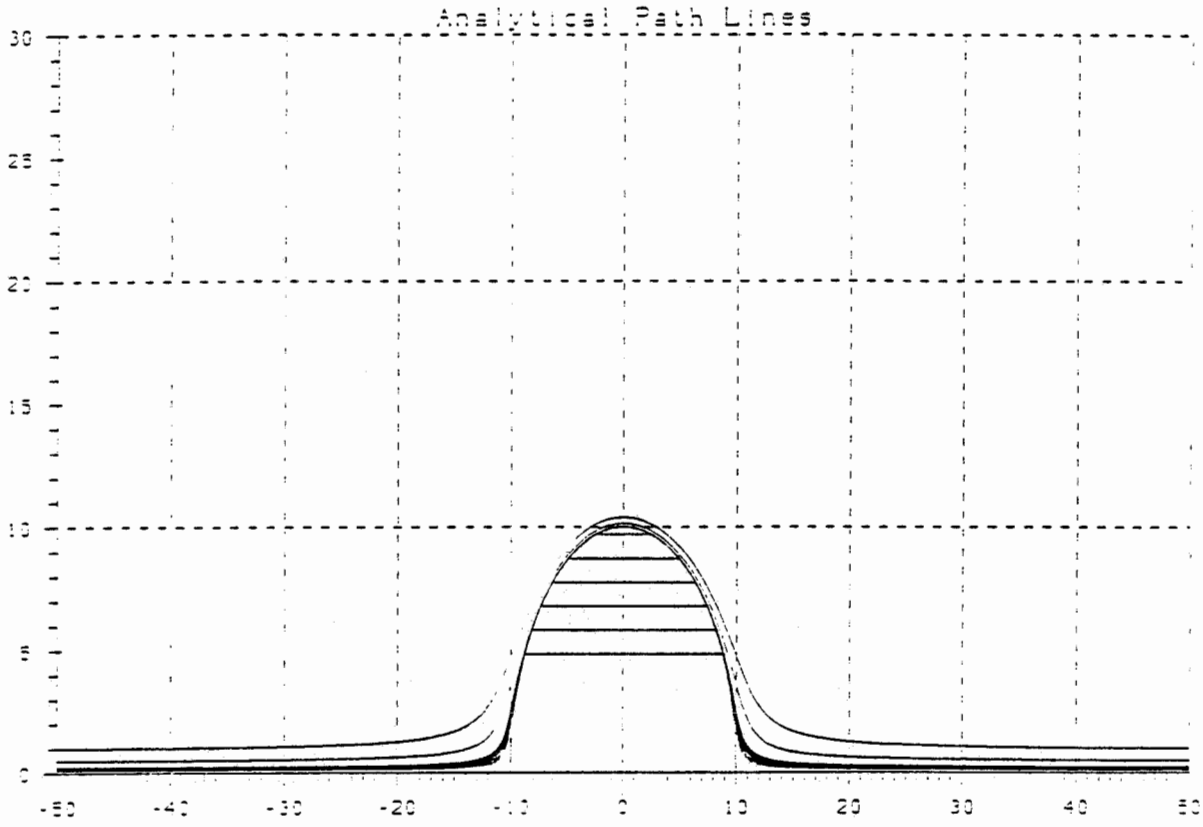
Die Potentiale werden an den Knoten analytisch vorgegeben. Beide Netze ergeben recht gute Ergebnisse. Im groben Netz scheinen einige Elemente noch zu stark gekrümmt zu sein. Bei einer Trajektorie bricht der Inversionsalgorithmus mit schlecht konditionierter Funktionalmatrix ab.

iv) Fall 4: Numerisch berechnetes Potential

Die Potentiale werden an den Knoten mit FEM301 berechnet. Beide Netze ergeben recht gute Resultate. Alle Trajektorien werden zu Ende gerechnet. Wegen der Stetigkeit des Potentials bereitet auch die Uebergangszone keine Schwierigkeiten.

5.1.8 Undurchlässiger Zylinder

i) Analytische Resultate

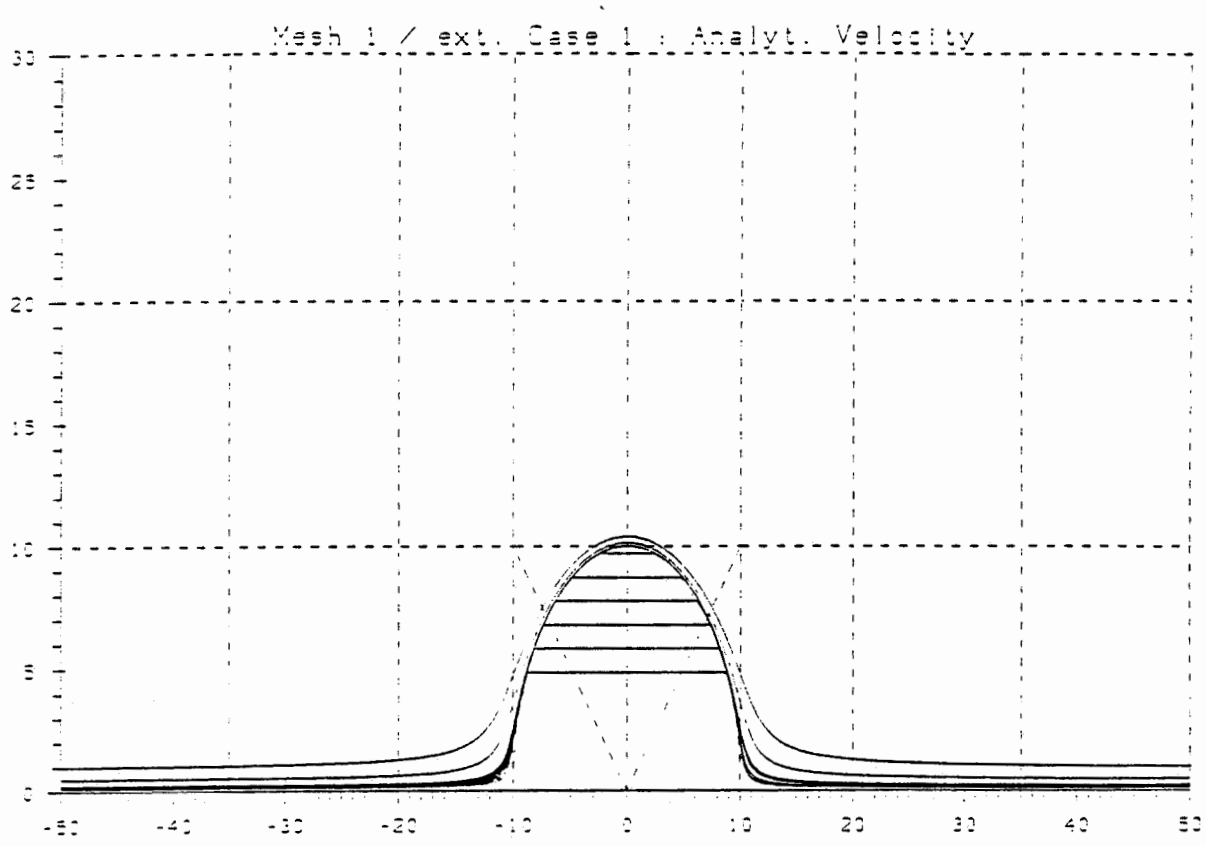


Units : Length [m] Time 1E09 [s]

Trg No	Y (X=0.)	Y (X=50.)	Time (X=0.)	Time (X=50.)
1	4.8520008	.1000000	50.6148804	101.2297607
2	5.8224014	.1200000	47.5319861	95.0639722
3	6.7928022	.1400000	43.5584281	87.1168562
4	7.7632032	.1600000	38.3611760	76.7223520
5	8.7336046	.1800000	31.1733240	62.3466480
6	9.7040063	.2000000	18.8636821	37.7273643
7	10.0067463	.2200000	6.7451184	13.4902368
8	10.0164615	.2400000	6.7020869	13.4041738
9	10.1436074	.5000000	6.3395332	12.6790664
10	10.3925474	1.0000000	5.9989825	11.9979649

Fig. 6: analytische Trajektorien

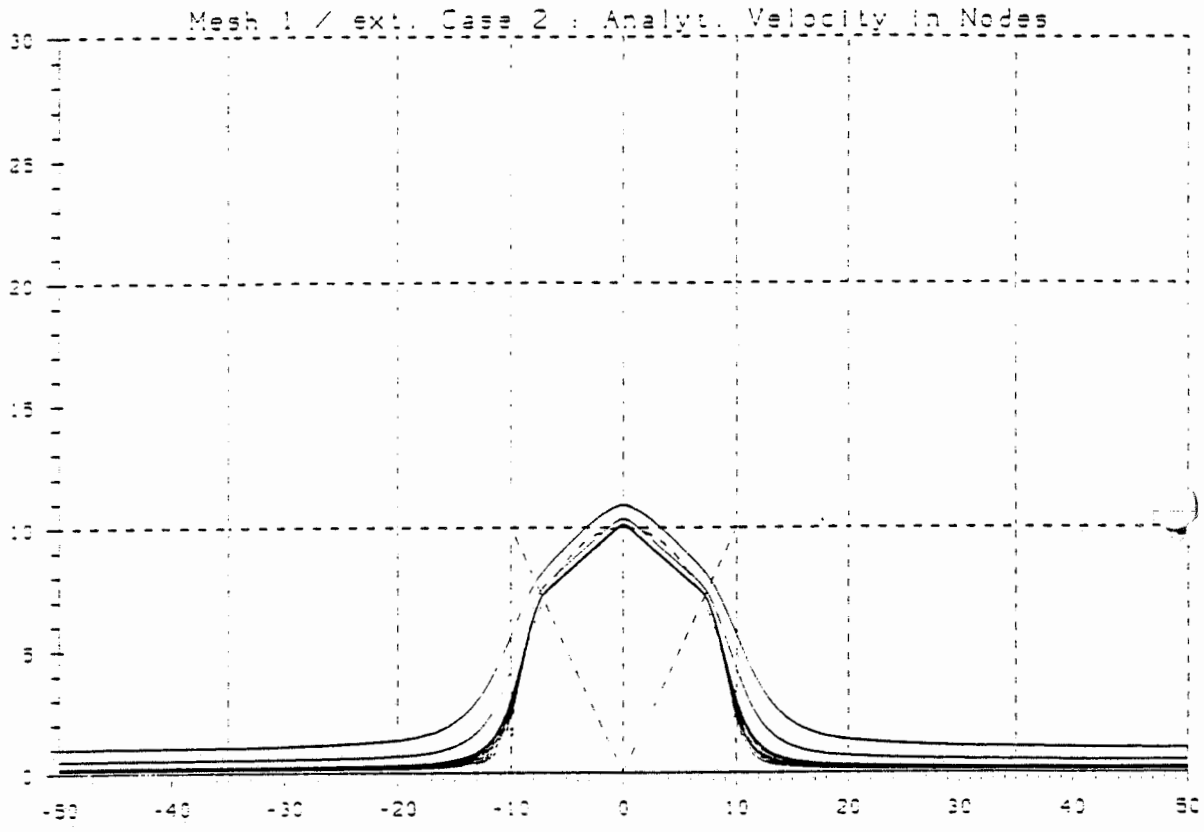
ii) Grobes Netz



Units : Length 1E+01 [m] Time 1E+11 [s]

Trg.No	1	2	3	4	5	6	7	8	9	10
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	0100001	0120001	.0140001	.0160001	.0180001	.0200001	.0220001	.0240001	.0500001	.1000001
Y (X=0)	4849111	.5827481	.6786041	.7759061	.8733841	9704741	1.0006651	1.0016381	1.0143521	1.0392461
Err (YO)	-.0002891	.0005081	-.0006771	-.0004141	.0000231	.0000731	-.0000091	-.0000081	-.0000081	-.0000091
X Final	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final	.0093491	.0111291	.0132891	.0134111	.0144131	.0197301	.0219601	.0239661	.0499651	.0999661
Err (YF)	-.0006511	-.0008711	-.0007111	-.0025891	-.0035871	-.0002701	-.0000401	-.0000341	-.0000351	-.0000341
T (X=0)	.5062241	.4751461	.4359131	.3838771	.3117211	.1885091	.0674531	.0670241	.0633971	.0599911
Err (TO)	.0000761	-.0001741	.0003291	.0002661	-.0000121	-.0001281	.0000021	.0000031	.0000011	.0000011
T Final	1.0350731	.9888981	.9064491	.8596151	.7706971	.4267391	.1349141	.1340531	.1267961	.1199821
Err (TF)	.0227761	.0382581	.0352801	.0923911	.1472301	.0494651	.0000121	.0000111	.0000051	.0000031

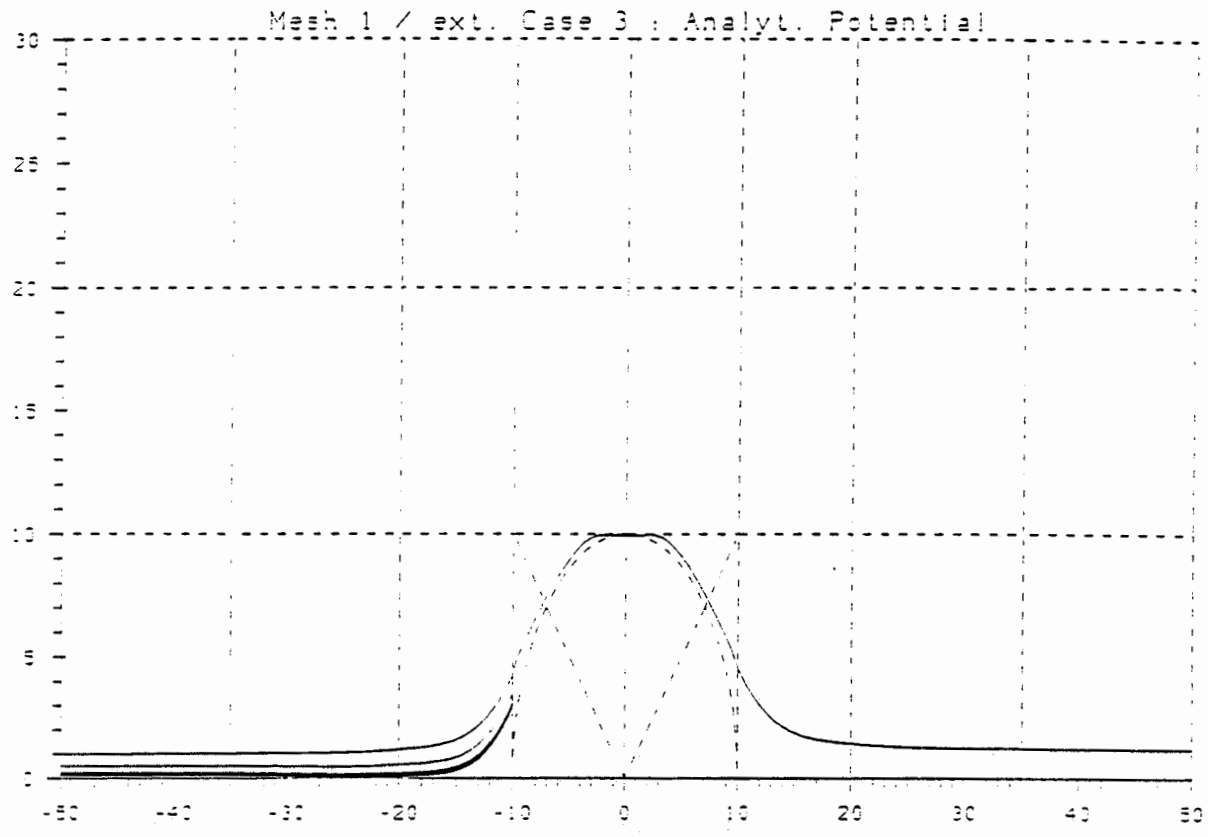
Fig. 7a: analytisches Geschwindigkeitsfeld



Units : Length 1E+01 [m] Time 1E+10 [s]

Tra.No	1	2	3	4	5	6	7	8	9	10
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	.0100001	.0120001	.0140001	.0160001	.0180001	.0200001	.0220001	.0240001	.0500001	.1000001
Y (X=0)	1.0089541	1.0103271	1.0109781	1.0114461	1.0121581	1.0130961	1.0139381	1.0149221	1.0358601	1.0925271
Err (Y0)	.5237541	.4280871	.3316981	.2351261	.1387971	.0426951	.0132631	.0132761	.0214991	.0532721
X Final	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final	.0095361	.0115701	.0135761	.0150241	.0170631	.0195401	.0215701	.0234301	.0498741	.0999771
Err (YF)	-.0004641	-.0004301	-.0004241	-.0009761	- 0009371	- 0004601	-.0004301	-.0005701	-.0001261	- 0000231
T (X=0)	.7741741	.7614451	.7538531	.7457881	.7380001	.7304791	.7249181	.7181631	.6557801	.5979901
Err (T0)	-4.2873141	-3.9917531	-3.6019901	-3.0903301	-2.3793331	-1.1558891	.0504061	.0479541	.0218271	-.0019091
T Final	1.5524161	1.5260661	1.5097141	1.4957721	1.4798061	1.4633511	1.4514551	1.4384591	1.3118671	1.1959881
Err (TF)	-8.5705601	-7.9803311	-7.2019721	-6.1764641	-4.7548591	-2.3093851	.1024311	.0980411	.0439601	-.0038091

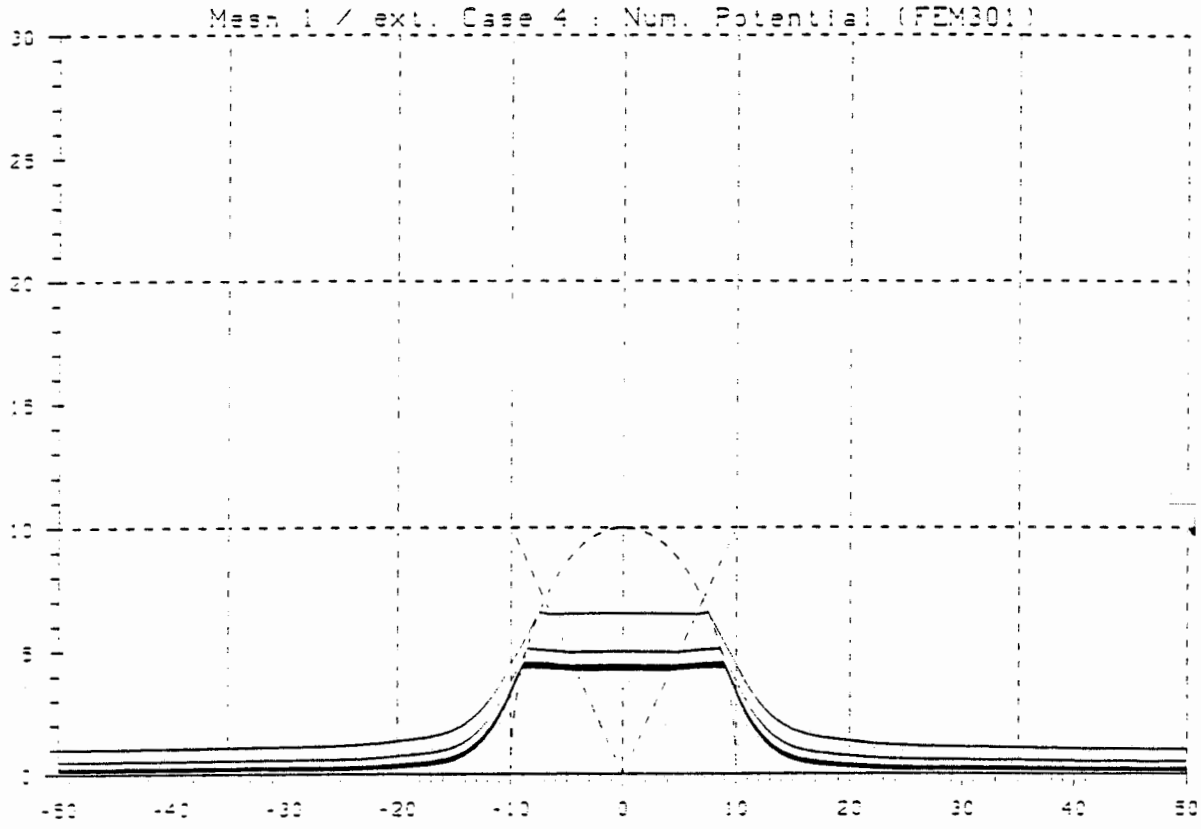
Fig. 7b: numerisches Geschwindigkeitsfeld



Units : Length 1E+01 [m] Time 1E+10 [s]

Trg.No	1	2	3	4	5	6	7	8	9	10
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	.0100001	.0120001	.0140001	.0160001	.0180001	.0200001	.0220001	.0240001	.0500001	.1000001
Y (X=0)	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	.9925411
Err (Y0)	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	- 0467141
X Final	-1.0000001	-1.0000001	-1.0000001	-1.0000001	-1.0000001	-1.0000001	-1.0000001	-1.0000001	-1.0000001	5.0000001
Y Final	.2912281	.2942581	.2972871	.3003141	.3033401	.3063651	.3093881	.3124101	.3515671	.1192441
Err (YF)	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	.0192441
T (X=0)	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	1.1840911
Err (T0)	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	.5841931
T Final	.5322011	.5316531	.5311091	.5305691	.5300321	.5295001	.5289711	.5284461	.5219381	2.3601291
Err (TF)	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	0.0000001	1.1603321

Fig. 7c: analytisches Potential

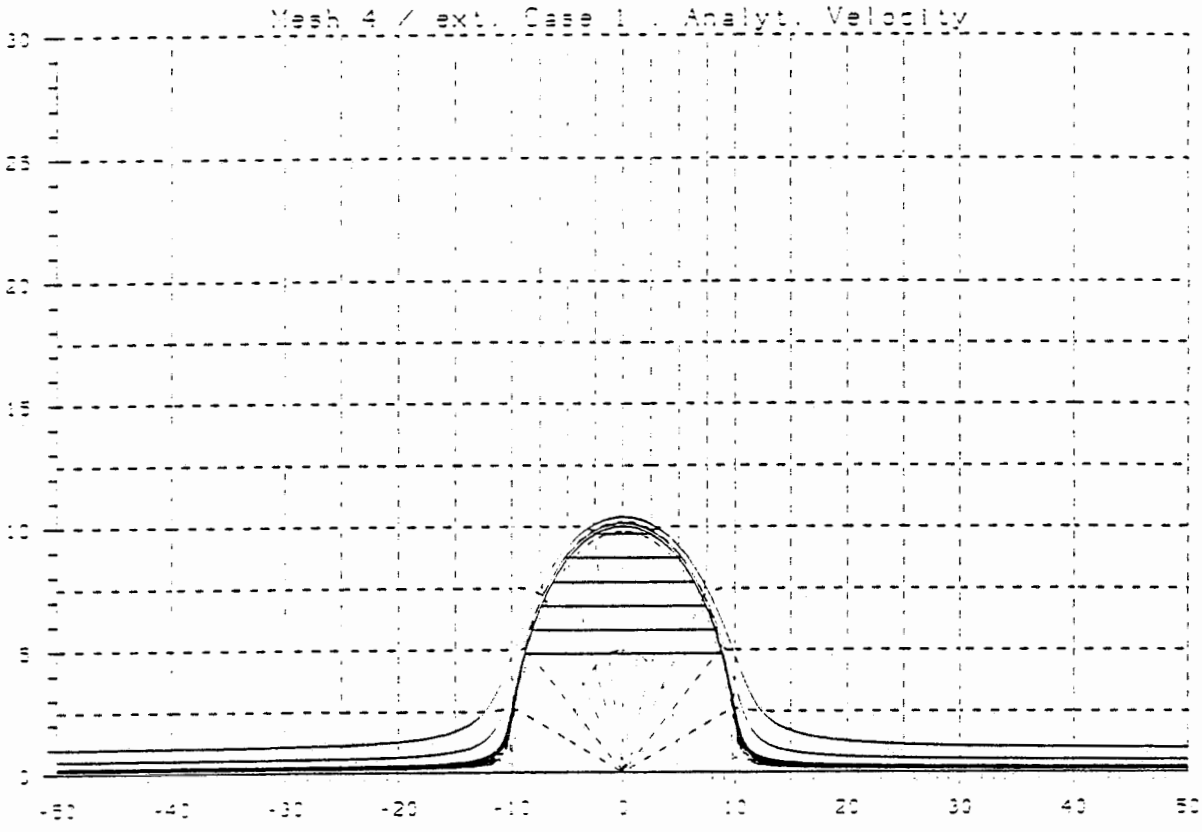


Units : Length 1E+01 [m] Time 1E+11 [s]

Trg.No	1	2	3	4	5	6	7	8	9	10
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	.0100001	.0120001	.0140001	.0160001	.0180001	.0200001	.0220001	.0240001	.0500001	.1000001
Y (X=0)	.4210771	.4243871	.4277571	.4311811	.4346661	.4382071	.4418061	.4454691	.4987241	.6538921
Err (Y0)	-.0641231	-.1578531	-.2515241	-.3451401	-.4386951	-.5321931	-.6258691	-.7561771	-.5156371	-.3853631
X Final	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final	.0100131	.0120111	.0140161	.0160131	.0180131	.0200131	.0220111	.0240131	.0500081	.1000131
Err (YF)	.0000131	.0000111	.0000161	.0000131	.0000131	.0000131	.0000111	.0000131	.0000081	.0000131
T (X=0)	.5190001	.5180831	.5171511	.5161921	.5152061	.5142011	.5131651	.5121071	.4956411	.4362771
Err (T0)	.0128521	.0427641	.0815671	.1325801	.2034731	.3255641	.4457141	.4450871	.4322461	.3762871
T Final	1.0379971	1.0361661	1.0342991	1.0323791	1.0304111	1.0283971	1.0263301	1.0242101	.9912791	.8725501
Err (TF)	.0257001	.0855271	.1631301	.2651551	.4069451	.6511231	.8914281	.8901681	.8644891	.7525701

Fig. 7d: numerisches Potential

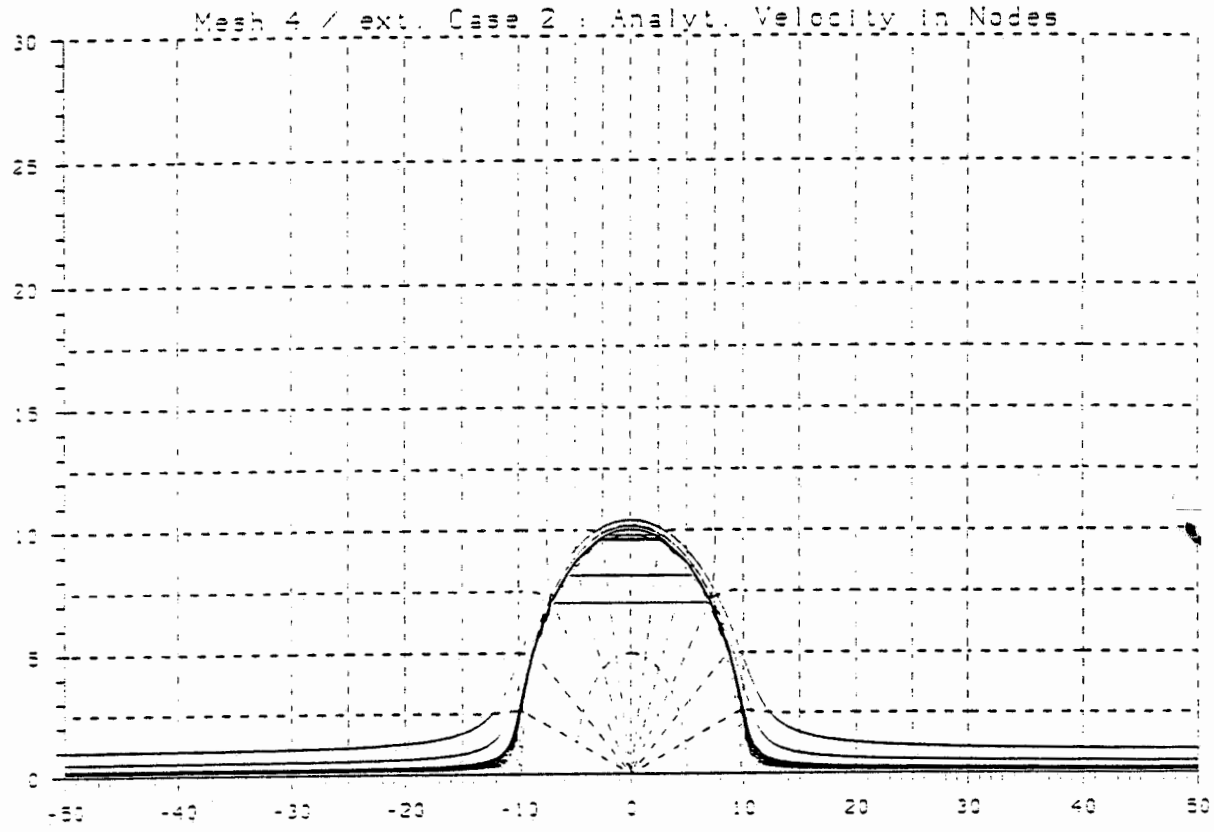
iii) Feines Netz



Units : Length 1E+01 [m] Time 1E+11 [s]

Trg.No	1	2	3	4	5	6	7	8	9	10
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	.0100001	.0120001	.0140001	.0160001	.0180001	.0200001	.0220001	.0240001	.0500001	.1000001
Y (X=0)	.4847251	.5818001	.6796451	.7767921	.8735731	.9704261	1.0006751	1.0016461	1.0143611	1.0392571
Err (Y0)	-.0004751	-.0004401	.0003651	.0004721	.0002121	.0000261	.0000001	.0000001	.0000011	.0000021
X Final	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final	.0099941	.0119761	.0139711	.0159781	.0180051	.0199581	.0220011	.0240011	.0500021	.1000061
Err (YF)	-.0000061	-.0000241	-.0000291	-.0000221	.0000051	-.0000421	.0000011	.0000011	.0000021	.0000061
T (X=0)	.5062721	.4754831	.4354161	.3833201	.3115431	.1885881	.0674511	.0670211	.0633951	.0599901
Err (T0)	.0001241	.0001631	-.0001681	-.0002911	-.0001911	-.0000491	.0000001	.0000001	.0000001	.0000001
T Final	1.0125741	.9509571	.8710141	.7671231	.6241891	.3796571	.1349021	.1340421	.1267911	.1199791
Err (TF)	.0002761	.0003181	-.0001541	-.0001011	.0007221	.0023841	-.0000001	.0000001	.0000001	-.0000001

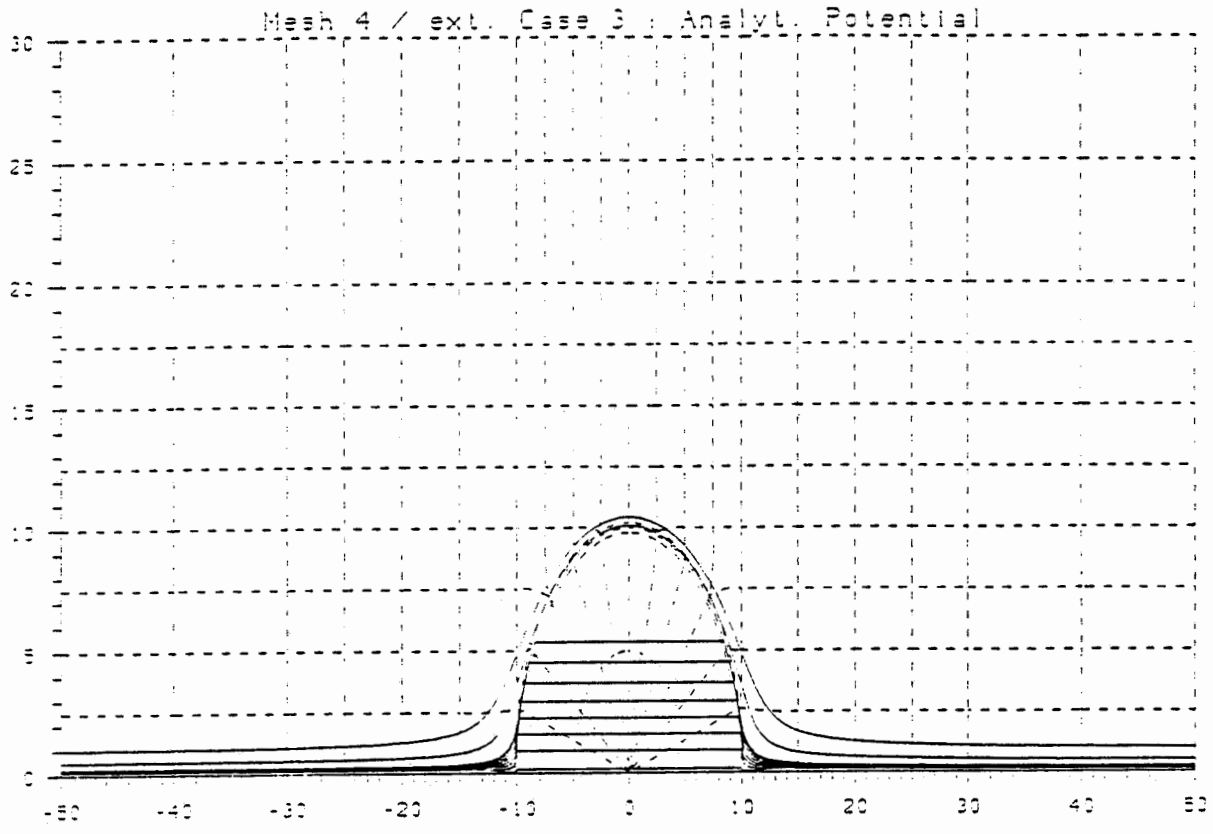
Fig. 8a: analytisches Geschwindigkeitsfeld



Units : Length 1E+01 [m] Time 1E+10 [s]

Trg.No	1	2	3	4	5	6	7	8	9	10
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	.0100001	.0120001	.0140001	.0160001	.0180001	.0200001	.0220001	.0240001	.0500001	.1000001
Y (X=0)	.7030771	.8153781	.9582841	.9644091	.9789581	1.0005911	1.0022611	1.0034011	1.0162201	1.0404511
Err (Y0)	.2178771	.2331381	.2790041	.1880881	.1055971	.0301901	.0015861	.0017551	.0018591	.0011971
X Final	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final	.0099951	.0119981	.0139921	.0159611	.0179801	.0199851	.0219941	.0239921	.0500011	.1000031
Err (YF)	-.0000051	-.0000021	-.0000081	-.0000391	-.0000201	-.0000151	-.0000061	-.0000081	.0000011	.0000031
T (X=0)	4.2330191	3.5233771	2.0668201	2.0168871	1.7359191	.7395941	.6908391	.6750731	.6310831	.6001031
Err (T0)	-.8284691	-1.2298221	-2.2890231	-1.8192311	-1.3814131	-1.1467741	.0163271	.0048641	-.0028701	.0002051
T Final	8.4660261	7.0466561	4.1331411	4.0327651	3.4724131	1.4792311	1.3816921	1.3501641	1.2621631	1.2002041
Err (TF)	-1.6569501	-2.4597411	-4.5785441	-3.6394701	-2.7622521	-2.2935051	.0326681	.0097461	-.0057441	.0004081

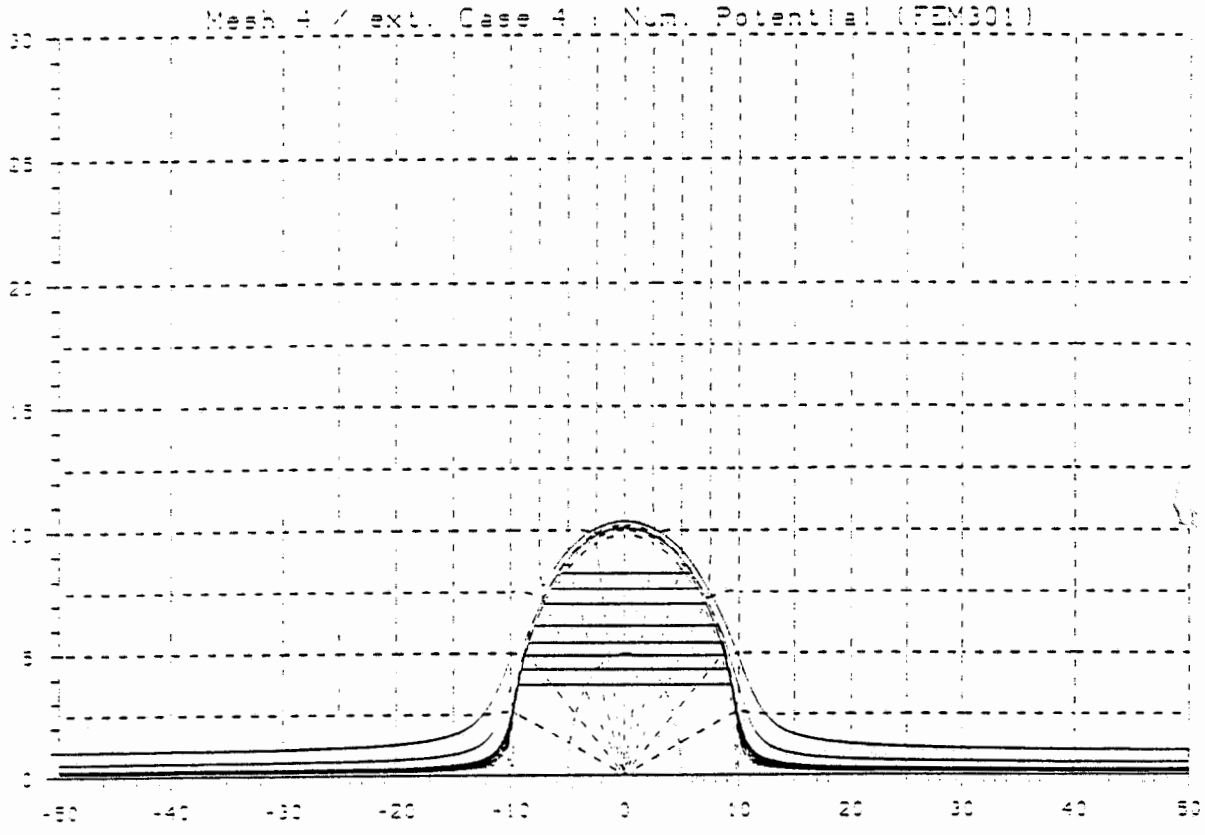
Fig. 8b: numerisches Geschwindigkeitsfeld



Units : Length 1E+01 [m] Time 1E+11 [s]

Trg.No	1	2	3	4	5	6	7	8	9	10
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	.0100001	.0120001	.0140001	.0160001	.0180001	.0200001	.0220001	.0240001	.0500001	.1000001
Y (X=0)	.0160131	.0919241	.1593571	.2252751	.2935621	.3686221	.4526661	.5367081	1.0084901	1.0439641
Err (Y0)	-.4691871	-.4903171	-.5199241	-.5510451	-.5797991	-.6017791	-.5480081	-.4649381	-.0058711	.0047091
X Final	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final	.0100001	.0120011	.0140011	.0160011	.0180011	.0200021	.0220011	.0240011	.0500041	.1000081
Err (YF)	.0000001	.0000011	.0000011	.0000011	.0000011	.0000021	.0000011	.0000011	.0000041	.0000081
T (X=0)	.5679501	.5651981	.5605281	.5538331	.5445651	.5313941	.5124711	.4885101	.0635071	.0599121
Err (T0)	.0618011	.0898781	.1249431	.1702221	.2328321	.3427571	.4450201	.4214891	.0001121	-.0000781
T Final	1.1359111	1.1303911	1.1210471	1.1076611	1.0891211	1.0627801	1.0249311	.9770141	.1270141	.1198231
Err (TF)	.1236131	.1797511	.2498791	.3404381	.4656551	.6855061	.8900281	.8429721	.0002231	-.0001571

Fig. 8c: analytisches Potential



Units : Length 1E+01 [m] Time 1E+11 [s]

Trg.No	1	2	3	4	5	6	7	8	9	10
X Start	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001	-5.0000001
Y Start	.0100001	.0120001	.0140001	.0160001	.0180001	.0200001	.0220001	.0240001	.0500001	.1000001
Y (X=0)	.3718781	.4348781	.4912091	.5433801	.6130401	.7014301	.7635851	.8265571	1.0097141	1.0361761
Err (Y0)	-.1133221	-.1473621	-.1880711	-.2329401	-.2603211	-.2689711	-.2370901	-.1750891	-.0046461	-.0030791
X Final	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001	5.0000001
Y Final	.0100021	.0120021	.0140011	.0160011	.0180021	.0200011	.0220021	.0240021	.0500041	.1000091
Err (YF)	.0000021	.0000021	.0000011	.0000011	.0000021	.0000011	.0000021	.0000021	.0000041	.0000091
T (X=0)	.5315691	.5174641	.5024741	.4868071	4622401	.4231671	3889931	.3476961	.0632941	.0599751
Err (T0)	.0254201	.0421441	.0668901	.1031951	.1505071	.2345311	.3215421	.2806751	- 0001011	-.0000151
T Final	1.0631291	1.0349161	1.0049401	.9736071	.9244721	.8463291	.7779811	.6953861	.1265881	.1199501
Err (TF)	.0508321	0842771	.1337711	.2063841	.3010051	.4690551	.6430791	.5613441	-.0002031	-.0000301

Fig. 8d: numerisches Potential

5.1.9 Diskussion der Resultate (undurchlässiger Zylinder)

i) Fall 1: Analytisches Geschwindigkeitsfeld

Die Qualität der Resultate ist vergleichbar mit dem vorherigen Beispiel.

ii) Fall 2: Diskretisiertes, analytisches Geschwindigkeitsfeld

Die Resultate sind auch in diesem Beispiel frappant. Für das feine Netz scheinen die Ergebnisse auf den ersten Blick recht mager. Eine Fehleranalyse (s. unten) zeigt, dass die grossen Diskrepanzen im Fehler der Aufenthaltszeiten im Zylinder zu suchen sind. Falls der Eintrittspunkt in den Zylinder (i.e. y-Komponente bei $x = 0$) wenig verschoben ist, resultieren recht unterschiedliche Zeiten.

Die Fehleranalyse basiert auf der Annahme, dass die Zeiten innerhalb des Zylinders der y-Positon gemäss den analytischen Werten entspricht. Vergleicht man nur die Zeiten ausserhalb des Zylinders, resultieren recht gute Ergebnisse. Den analytischen Wert der Aufenthaltszeit innerhalb des Zylinders erhält man gemäss Formel (5-5) aus $\vec{v}$ für $r < a$ und der Länge des Weges.

Fehleranalyse

	Y0 [m]	TF [1E10 s]	TC [1E10 s]	TB [1E10 s]
Numerical	9.583	4.133	2.886	1.247
Analytical	6.793	8.712	7.412	1.3
Err (N-A)	2.79	-4.579	-4.526	-0.053

Table : ERROR ANALYSIS OF MESH 4 / CASE 2 : Track No 3

Y0 : Y at X=0 ; i.e. Y-Location of Entrance into Inclusion
TF : Cumulative Time at X=50 m
TC : Analytical Travel Time Spent within The Inclusion Calculated Using Y0 and Analytical Velocity
TB : Difference (TF-TC) ; i.e. Travel Time within Bulk

iii) Fall 3: Analytisches Potentialfeld

Die Ergebnisse sind vergleichbar mit Fall 2. Der Abbruch einiger Trajektorien wegen schlecht konditionierter Elemente kommt im groben Netz deutlich zum Ausdruck. Eine Fehleranalyse zeigt, dass auch in diesem Fall die Verschiebung des Eintrittspunktes die Hauptfehlerquelle ist.

Fehleranalyse

	YO [m]	TF [1E10 s]	TC [1E10 s]	TB [1E10 s]
Numerical	3.686	10.627	9.389	1.238
Analytical	9.704	3.773	2.439	1.334
Err (N-A)	-6.018	6.854	6.95	-0.096

Table : ERROR ANALYSIS OF MESH 4 / CASE 3 : Track No 6

iv) Fall 4: Numerisch berechnetes Potential

Die auf den numerischen Potentialen beruhenden Resultate sind vergleichbar mit Fall 3, wobei exaktere Werte und Verläufe festzustellen sind. Ein Erklärungsversuch wird anschliessend besprochen. Die Fehleranalyse wiedergibt den bereits festgestellten Sachverhalt.

Fehleranalyse

	YO [m]	TF [1E10 s]	TC [1E10 s]	TB [1E10 s]
Numerical	7.014	8.463	7.199	1.264
Analytical	9.704	3.773	2.439	1.334
Err (N-A)	-2.69	4.69	4.76	-0.07

Table : ERROR ANALYSIS OF MESH 4 / CASE 4 : Track No 6

5.1.10 Folgerungen

Aus den beobachteten Vergleichen sind einige Schlüsse zu ziehen.

Der Hochdurchlässige Zylinder bietet fast keine Schwierigkeiten. Achtung ist geboten, wenn unstetige Felder (Geschwindigkeitsfeld, Potentialgradient etc.) diskretisiert werden. Starke Kontraste bedürfen einer sorgfältigen Diskretisierung. Ebenfalls muss bei gekrümmten Elementen deren Regularität im Feinen geprüft werden.

Die Umströmung des undurchlässigen Zylinders wirft, wie zu erwarten war, mehr Probleme auf. Da nur die Trajektorien in der Nähe der horizontalen Symmetrieachse den Zylinder durchqueren, sind die Krümmungen der Linien in der Uebergangszone recht ausgeprägt. Es wird erwartet, dass eine problemangepasstere Diskretisierung in diesem Bereich zu besseren Resultaten führt. Damit kann auch die bessere Qualität der auf den numerischen Potentia-

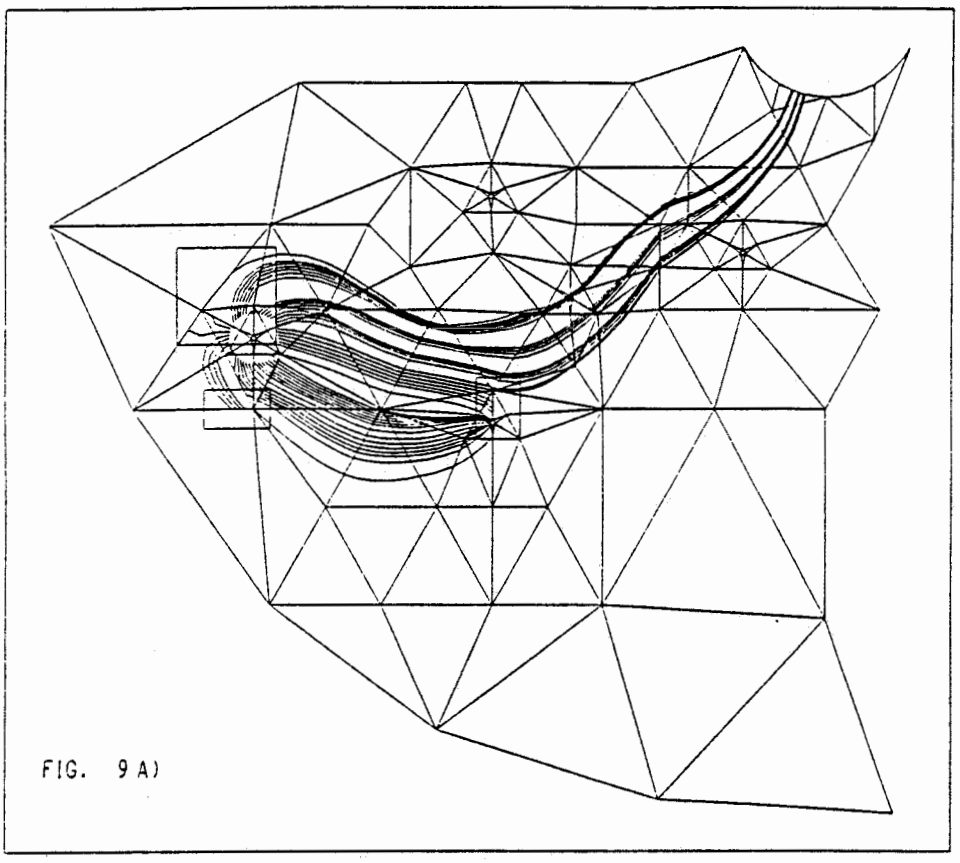
len beruhenden Berechnungen im Vergleich zu den analytischen Potentialen erklärt werden. Die Elemente im Krümmungsbereich sind zu grob, um den exakten Verlauf einigermassen genau wiedergeben zu können. Abschliessend soll zur Rechtfertigung doch noch erwähnt sein, dass die gewählten Netze dem Problem des durchlässigen Zylinders angepasst wurden.

5.2 Operationelles Beispiel

Der vorliegende Fall beschreibt einen Ausschnitt aus einem 2-dimensionalen Modell des Felslabors Grimsel. Eine detaillierte Beschreibung ist dem Bericht "Hydraulisches Modell für die Migrationszone im Felslabor Grimsel" ([redacted] 1988) zu entnehmen. Gezeigt werden Trajektorien, die bei Injektion in ein Bohrloch beginnen (kreisförmige Anordnung der Startpunkte links im Bild) und teils in den Felslaborstollen (rechts oben im Bild) und teils in ein geöffnetes Bohrloch (ca. Mitte des Bildes) migrieren.

Das Beispiel wurde mit einem anderen Programmcode nachgerechnet (NAMMU-NAMSOL von Harwell, GB) und die Resultate verglichen. Die Ergebnisse zeigen befriedigende Uebereinstimmung der beiden Codes. Einige Unterschiede werden nachfolgend illustriert, insbesondere das Auftreten von Oszillationen.

3 Ausschnitte - durch Rechtecke markiert - sind vergrössert dargestellt. Sie verdeutlichen, wie das Programm Track auftretende Oszillationen an Elementrändern handhabt. Durch die Wahl des grössten Geschwindigkeitsvektors entlang des Randes wird die Aufenthaltszeit der Trajektorie minimiert. Oszillationen über eine längere Wegstrecke können merkliche Unterschiede in den Laufzeiten bewirken (wurde in diesem Beispiel nicht festgestellt). Die exakte Bestimmung der Austrittskoordinaten aus einem Element verhindert ein Ueberschneiden der Trajektorien. Die in der Vergrösserung sichtbare grosse Schrittweite (zackige Linie) wird durch eine Input-Option gesteuert, die ein Filtern der Trajektorienkoordinaten bewirkt, um "handliche" Output-Daten zu erhalten.



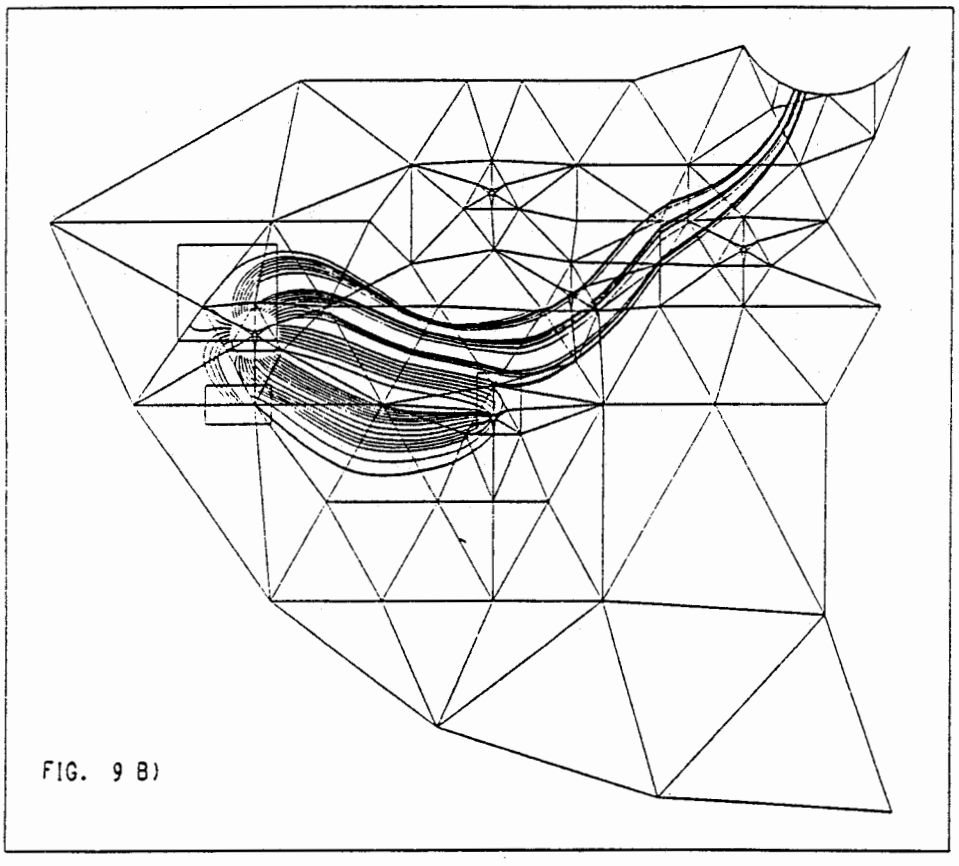
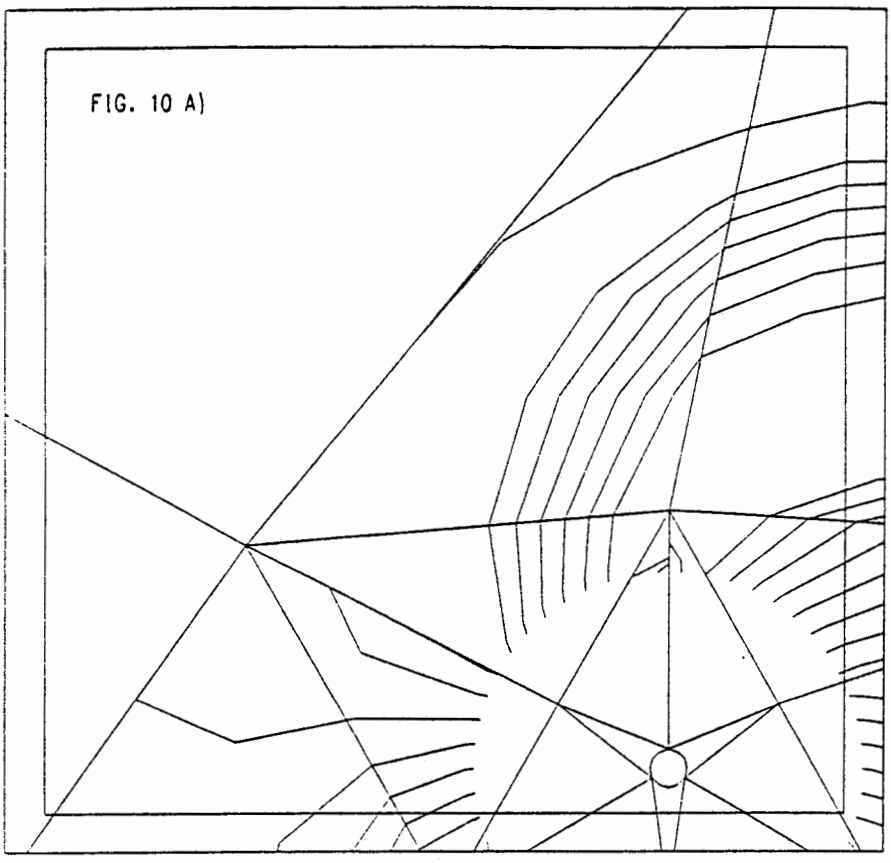


Fig. 9: a) TRACK-Trajektorien, b) NAMMU-Trajektorien



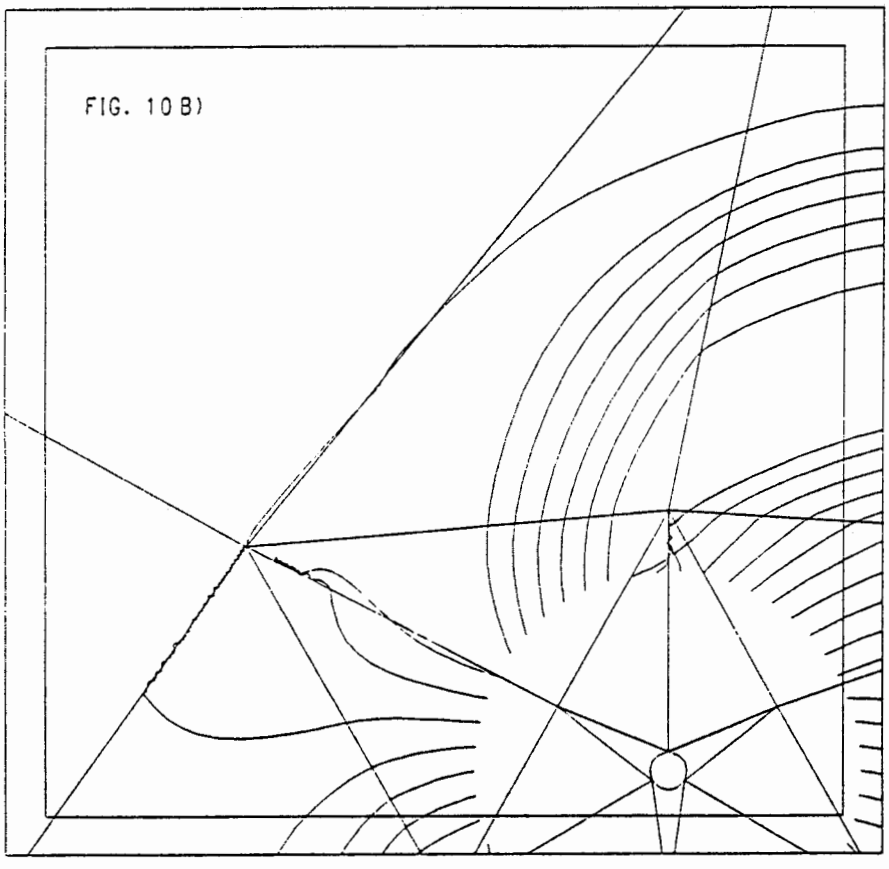


Fig. 10: Ausschnitt oben links: a) TRACK, b) NAMMU

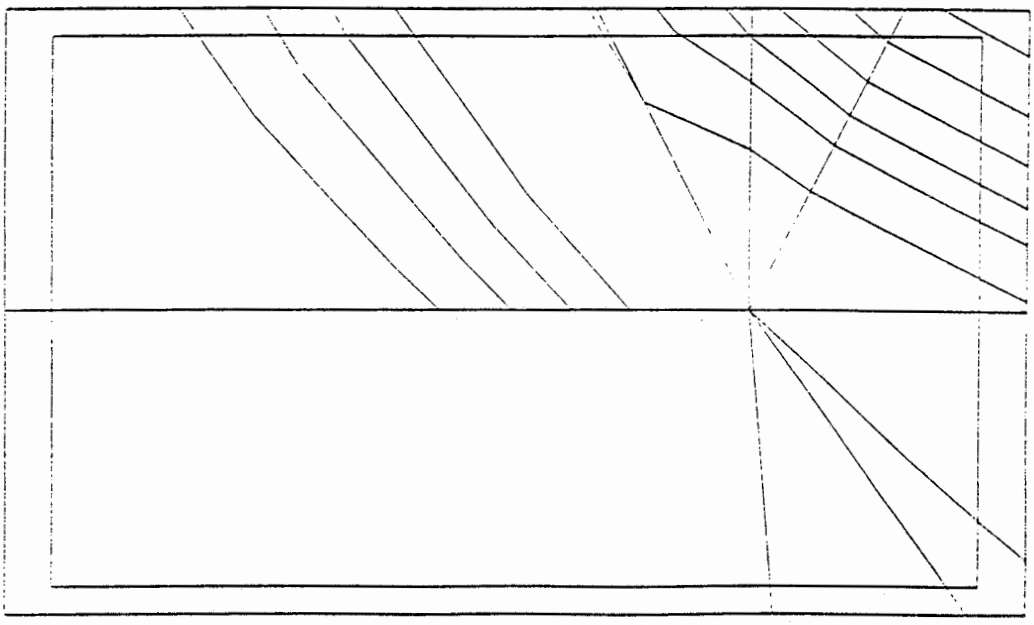


Fig. 11a: TRACK: Ausschnitt unten links

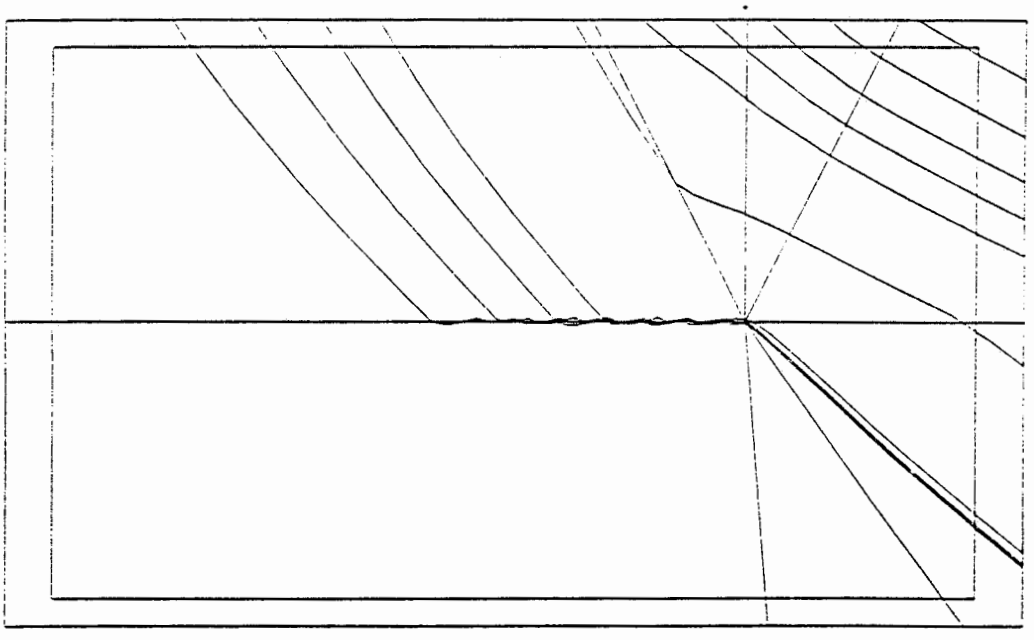


Fig. 11b: NAMMU: Ausschnitt unten links

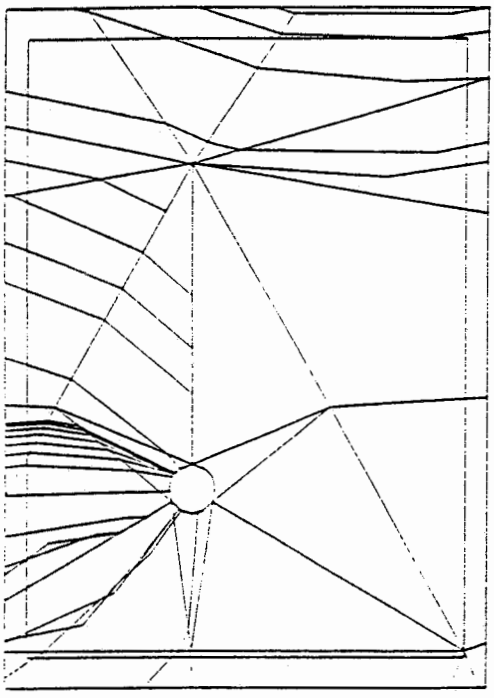


Fig. 12a: TRACK: Ausschnitt Mitte

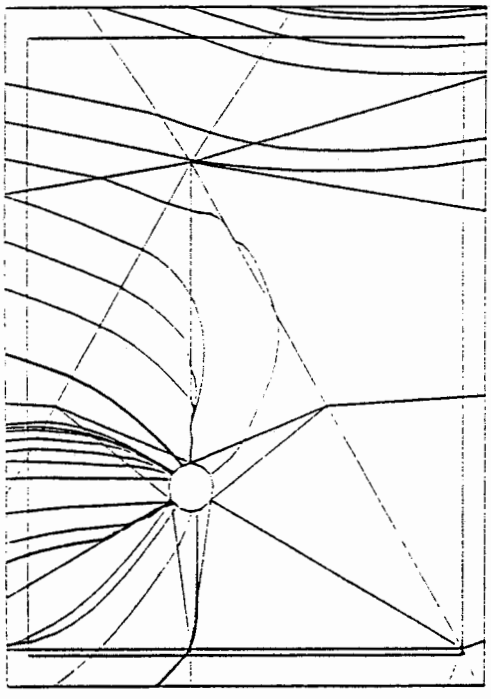


Fig. 12b: NAMMU: Ausschnitt Mitte

6. LITERATURVERZEICHNIS

Bronstein-Semendjajew (1979): Taschenbuch der Mathematik.
Verlag Harri Deutsch

Jackson, C.P. (1987): Proposal for an extension to HYDROCOIN Level 3 Case 7: A test for particle tracking algorithms.
HYDROCOIN 87 (7)

Kiraly, L. (1985): FEM301-A three Dimensional Model for Groundwater Flow Simulation.
NTB 84-49

Stiefel, E. (1980): Methoden der mathematischen Physik.
VdF, Zürich

Zienkiewicz, O.C. (1984): Methode der finiten Elemente.
Hauser Verlag, München

ANHANG

A 1. Programmlisting

```
      PROGRAM TRACK
C-----------------------------------------------------------------------
C PARTICLE TRACKING
C-----------------------------------------------------------------------

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL CDC
      PARAMETER       (CDC=.TRUE.)

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
      COMMON /ORIGIN/ XORIG(3)



      CHARACTER*256 FIELM,FICOR,FITRA,FIPAR,FIPAT,FILVT,FIPOT,
     &              FISTA,FICON,FITAB
      CHARACTER    USER*10,DTM*16,VERS*5,DATU*11
      LOGICAL NEWCON,ICON2

      VERS='2.3'
      DATU='06-OCT-1988'

      INELM=7
      INCOR=8
      INPAR=9
      OUTRA=10
      INPOT=11
      INSTA=12
      OUPAT=13
      IOMAT=14
      OULVT=15
      OUTAB=16


C Initialize NOS/VE Terminal I/O
      IF(CDC) THEN
         CALL SETTERM(-1,-2)
         CALL TERMIO(LI,LO)
      ENDIF

      CALL DAYTIM(DTM)
      CALL PROMPT(' USER : ')
      READ(1,'(A)') USER
      CALL CPUTIM(OUTRA,'INIT',USER)
      DO 10 I=1,IV1
10    POT(I)=0D0
      DO 20 I=1,IV3
      PORV(I)=0D0
20    PERA(1,I)=-1D0

      CALL OLDFILE(INCOR,' COORDINATE INPUT FILE      :      ',
     &             FICOR)
      CALL OLDFILE(INELM,' ELEMENT INPUT FILE         :      ',
     &             FIELM)
      CALL OLDFILE(INPAR,' PARAMETER INPUT FILE       :      ',
     &             FIPAR)

C Read potential
```

```
30    CALL OLDFILE(INPOT,' POTENTIAL INPUT FILE      :      ',
     &             FIPOT)
      CALL FINDKW(INPOT,'RESULTS',NB)
      IF(NB.LT.1) GOTO 30
      CALL R8HEAD(INPOT,POT,IV1)
      CALL CLOSEF(INPOT)

      CALL OLDFILE(INSTA, ' START POINTS INPUT FILE   :      ',
     &             FISTA)
      CALL NEWFILE(OUTRA,' TRACKING RESULT FILE  : '//
     &                   '[DEF=TRACK_TRA] : ','TRACK_TRA',FITRA)

      CALL NEWFILE(OUPAT,' PATH OUTPUT FILE  : '//
     &                   '[DEF=TRACK_PAT] : ','TRACK_PAT',FIPAT)

      CALL NEWFILE(OULVT,' LENGHT VS TIME OUTPUT FILE  : '//
     &                   '[DEF=TRACK_LVT] : ','TRACK_LVT',FILVT)

      CALL NEWFILE(OUTAB,' TABLE OF RESULTS OUTPUT FILE  : '//
     &                   '[DEF=TRACK_TAB] : ','TRACK_TAB',FITAB)
      WRITE(OUTAB,'(A,//A,/A)') FITAB,
     &'TRACK NO.  LAENGE (m)     ZEIT (y)     GESCHW. (m/y)'//
     &'   MESSAGE',
     &'-------------------------------------------------'//
     &'----------------------------------------'
      NEWCON=.TRUE.
      CALL QUEST(' Create new connection matrix ? [def=Y]',NEWCON)
      IF(NEWCON) THEN
         CALL NEWFILE(IOMAT,' CONNECTION MATRIX OUTPUT FILE : '//
     &                      '[DEF=TRACK_CON] :','TRACK_CON',FICON)
      ELSE
50       CALL OLDFILE(IOMAT,' CONNECTION MATRIX INPUT FILE :  ',
     &               FICON)
         CALL FINDKW(IOMAT,'CONNECT2',NBC)
         IF (NBC.LT.1) THEN
           CALL FINDKW(IOMAT,'CONNECTION',NBC)
           IF(NBC.LT.1) GOTO 50
           PRINT*,'CONNECTION MATRIX VERSION 1'
           ICON2=.FALSE.
         ELSE
           PRINT*,'CONNECTION MATRIX VERSION 2'
           ICON2=.TRUE.
         ENDIF
      ENDIF

      WRITE(OUTRA,'(/A)')
     &     ' ************************************************'
      WRITE(OUTRA,'(A)')
     &     ' * PARTICLE TRACKING FOR FINITE ELEMENTS          *'
      WRITE(OUTRA,'(A)')
     &     ' * IN THREE DIMENSIONS                            *'
      WRITE(OUTRA,'(A)')
     &     ' * VERSION          '//VERS//'                          *'
      WRITE(OUTRA,'(A)')
     &     ' * ETH/VAW-MCI      '//DATU//'                       *'
      WRITE(OUTRA,'(A)')
     &     ' * TIME: '//DTM//'     USER: '//USER//'       *'
      WRITE(OUTRA,'(A)')
     &     ' ************************************************'

      CALL LENWOB(FIELM,LELM)
      CALL LENWOB(FICOR,LCOR)
      CALL LENWOB(FIPAR,LPAR)
      CALL LENWOB(FIPOT,LPOT)
      CALL LENWOB(FICON,LCON)
      CALL LENWOB(FISTA,LSTA)
      CALL LENWOB(FITRA,LTRA)
      CALL LENWOB(FIPAT,LPAT)
      CALL LENWOB(FILVT,LLVT)
      LELM=MAX(1,MIN(100,LELM))
      LCOR=MAX(1,MIN(100,LCOR))
      LPAR=MAX(1,MIN(100,LPAR))
      LPOT=MAX(1,MIN(100,LPOT))
      LSTA=MAX(1,MIN(100,LSTA))
      LCON=MAX(1,MIN(100,LCON))
```

```
      LTRA=MAX(1,MIN(100,LTRA))
      LPAT=MAX(1,MIN(100,LPAT))
      LLVT=MAX(1,MIN(100,LLVT))
      WRITE(OUTRA,'(/A)') '     Input files:  '//FIELM(:LELM)
      WRITE(OUTRA,'(A)') '                   '//FICOR(:LCOR)
      WRITE(OUTRA,'(A)') '                   '//FIPAR(:LPAR)
      WRITE(OUTRA,'(A)') '                   '//FIPOT(:LPOT)
      WRITE(OUTRA,'(A)') '                   '//FISTA(:LSTA)
      IF(.NOT.NEWCON) WRITE(OUTRA,'(A)') '                   '//
     &                                     FICON(:LCON)
      WRITE(OUTRA,'(A)')
      WRITE(OUTRA,'(A)') '    Output files: '//FITRA(:LTRA)
      WRITE(OUTRA,'(A)') '                   '//FIPAT(:LPAT)
      WRITE(OUTRA,'(A)') '                   '//FILVT(:LLVT)
      IF(NEWCON) THEN
         WRITE(OUTRA,'(A)') '                     '//FICON(:LCON)
         CALL SAVEHED(IOMAT,FICON,FIELM//FIPAR,'TRACK',VERS,DTM,USER)
      ENDIF
      CALL SAVEHED(OUPAT,FIPAT,FIELM//FICOR//FIPAR//FIPOT//FISTA,
     &             'TRACK',VERS,DTM,USER)
      CALL SAVEHED(OULVT,FILVT,FIELM//FICOR//FIPAR//FIPOT//FISTA,
     &             'TRACK',VERS,DTM,USER)

      IF (NEWCON) THEN
         CALL PROMPT(' Reading input files and '//
     &       'calculating CONNECTION MATRIX')
      ELSE
         CALL PROMPT(' Reading input files')
      ENDIF
      PRINT*
C Read permeabilities and infiltration rates
      CALL LECPE(IERR)
      IF (IERR.NE.0) GOTO 1000
C Read elements
      CALL LECELM(IERR)
      IF (IERR.NE.0) GOTO 1000
C Read coordinates
      CALL LECCOR(IERR)
      IF (IERR.NE.0) GOTO 1000
      IF (NEWCON)
     &   CALL CPUTIM(OUTRA,'READ INPUT FILES                ',USER)

C Calculate connection matrix ICON
      CALL CONMAT(NEWCON,FICON,ICON2,NBC,IERR)
      IF (IERR.NE.0) GOTO 1000
      IF(NEWCON) THEN
         CALL CPUTIM(OUTRA,'CALCULATE CONNECTION MATRIX     ',USER)
      ELSE
         CALL CPUTIM(OUTRA,'READ INPUT FILES                ',USER)
      ENDIF

C Calculate tracks and print results
      CALL STREAM(USER)
      CALL CPUTIM(OUTRA,'RUN "TRACK" TERMINATED',USER)

C End of run
 1000 CONTINUE

C Deactivate NOS/VE Terminal I/O, close files
      IF(CDC) THEN
         CALL OFFTERM(LI)
         CALL OFFTERM(LO)
      ELSE
         ENDFILE(OUTRA)
         ENDFILE(OUPAT)
         ENDFILE(OULVT)
         ENDFILE(OUTAB)
      ENDIF
      CALL CLOSEF(INSTA)
      CALL CLOSEF(OUTRA)
      CALL CLOSEF(OUPAT)
      CALL CLOSEF(OULVT)
```

```
      CALL CLOSEF(OUTAB)

      STOP
C----------END TRACK------------
      END


      SUBROUTINE LECCOR(IERR)

      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
      COMMON /ORIGIN/ XORIG(3)



C     Input of the global coordinates X,Y,Z.
C     --------------------------------------

      IERR=0
      CALL FINDKW(INCOR,'COORDINATES',NLINE)
      IF (NLINE.LT.1) THEN
        IERR=200
        RETURN
      ENDIF

C Get space dimension and scale factors

      CALL CNTWORD(INCOR,ND,IEND)
      IF(ND.LT.2 .OR. ND.GT.3) GOTO 900
      IF(ND.EQ.3) THEN
        READ(INCOR,*,ERR=901) FACX,FACY,FACZ
      ELSE
        READ(INCOR,*,ERR=901) FACX,FACY
        FACZ=0D0
      ENDIF
      NLINE=NLINE+1

C Read data

C First line: get XORIG(1) and XORIG(2) as new origin
      NLINE=NLINE+1
      IF (ND.EQ.3) THEN
         READ(INCOR,*,ERR=902,END=20) NIC,XX,YY,ZZ
      ELSE
         READ(INCOR,*,ERR=902,END=20) NIC,XX,YY
         ZZ=0D0
      ENDIF

      IF(NIC.GT.IV1) GOTO 903
      IF(IDCOR(NIC).GE.1) THEN
        XORIG(1)=XX*FACX
        XORIG(2)=YY*FACY
        XORIG(3)=0D0
        COOR(1,NIC)=0D0
        COOR(2,NIC)=0D0
```

```
        COOR(3,NIC)=ZZ*FACZ
        IDCOR(NIC)=IDCOR(NIC)+10000
      ENDIF

C Rest of data
10    NLINE=NLINE+1
      IF (ND.EQ.3) THEN
         READ(INCOR,*,ERR=902,END=20) NIC,XX,YY,ZZ
      ELSE
         READ(INCOR,*,ERR=902,END=20) NIC,XX,YY
         ZZ=0D0
      ENDIF

      IF(NIC.GT.IV1) GOTO 903
      IF(IDCOR(NIC).GE.1) THEN
        COOR(1,NIC)=XX*FACX-XORIG(1)
        COOR(2,NIC)=YY*FACY-XORIG(2)
        COOR(3,NIC)=ZZ*FACZ
        IDCOR(NIC)=IDCOR(NIC)+10000
      ENDIF
      GOTO 10

20    ISTOP=0
      DO 30 NIC=MNNIC,MXNIC
        IF(IDCOR(NIC).NE.0.AND.IDCOR(NIC).LT.10001) THEN
          WRITE(OUTRA,20004) NIC
          IERR=200
        ENDIF
        IDCOR(NIC)=IDCOR(NIC)-10000
        IF (IDCOR(NIC).GT.10000) THEN
          WRITE (OUTRA,20005) NIC
          IERR=200
        ENDIF
30    CONTINUE

      IF(IERR.NE.0) GOTO 910
      CALL CLOSEF(INCOR)
      RETURN

900   WRITE(OUTRA,20000)
      GOTO 910
901   WRITE(OUTRA,20001)
      GOTO 910
902   WRITE(OUTRA,20002) NLINE
      GOTO 910
903   WRITE(OUTRA,20003) NIC,NLINE
910   CONTINUE
      CALL CLOSEF(INCOR)
      IERR=200
      RETURN

20000 FORMAT(/' SPACE DIMENSION IN COORDINATE FILE',
     &        ' OUT OF RANGE !')
20001 FORMAT(/' FORMAT ERROR WHILE READING SCALE FACTORS',
     &        ' IN COORDINATE FILE !')
20002 FORMAT(/' FORMAT ERROR WHILE READING LINE ',I5,
     &        ' OF COORDINATE FILE !')
20003 FORMAT(/' NODE NUMBER ',I5,' OUT OF RANGE',
     &       /' CHECK LINE  ',I5,' OF COORDINATE FILE !')
20004 FORMAT(/' NO COORDINATES GIVEN FOR NODE ',I6,' !')
20005 FORMAT(/' COORDINATES FOR NODE ',I6,' GIVEN MORE THAN ONCE !')
C----------END LECCOR-----------
      END


      SUBROUTINE LECELM(IERR)

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL PINCH

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)
```

```
      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------

      DIMENSION LVB(27)

C     READ ELEMENT DATA
C     -----------------

      CALL FINDKW(INELM,'@PYRAMID=BRICK',NBLINE)
      PINCH=(NBLINE.GT.0)
      IERR=0
      CALL FINDKW(INELM,'ELEMENTS',NBLINE)
      IF (NBLINE.LT.1) THEN
        IERR=100
        RETURN
      ENDIF

      MAXDIM=0
      MXLM=0
      MNNIC=IV1
      MXNIC=0
      DO 10 I=1,IV1
   10 IDCOR(I)=0

  100 READ(INELM,*,END=400,ERR=500) LM,NQ,NP,KR,NAR,(LVB(K),K=1,KR)

      IF (PINCH) THEN
C get pinched brick out of pyramid
         IF (KR.EQ.13) THEN
            KR=20
            NAR=-8
            DO 150 K=14,20
  150       LVB(K)=LVB(13)
         ENDIF
      ENDIF

      IF (PERA(1,NP).LT.0D0) THEN
         WRITE(OUTRA,20001) NP,LM
         IERR=100
      ENDIF

C ELEMENT WITHOUT PERMEABILITY IS SKIPPED
      IF (.NOT.(PERA(1,NP)+PERA(3,NP)+PERA(6,NP).GT.0D0)) GOTO 100

      MAXDIM=MAX(MAXDIM,NDIM(KR))
      MXLM=MXLM+1
      DO 200 K=1,KR
        NIC=LVB(K)
        NELC(K,MXLM)=NIC
        INVELM(LM)=MXLM
        IDCOR(NIC)=IDCOR(NIC)+1
        IF(NIC.LT.MNNIC) MNNIC=NIC
        IF(NIC.GT.MXNIC) MXNIC=NIC
        IREP=1
        DO 180 KK=K+1,KR
          IF(NIC.NE.LVB(KK)) GOTO 180
          IREP=IREP+1
  180   CONTINUE
        IF(IREP.NE.1.AND.NAR.GT.0) THEN
          WRITE(OUTRA,700)LM,NIC,IREP
          IERR=100
        ENDIF
C         ------------------------------------------------------
```

```
  200 CONTINUE
      IELM(MXLM)=LM
      NELP(MXLM)=NP
      INVELM(LM)=MXLM
      KRV(MXLM)=KR
      NARV(MXLM)=NAR
      IF(KR .LT. 27 ) THEN
        DO 250 K=KR+1,27
  250   NELC(K,MXLM)=0
      ENDIF
      GOTO 100
  400 CONTINUE
      CALL CLOSEF(INELM)
      RETURN

500   WRITE (OUTRA,*) ' ERROR DURING READING OF ELEMENT FILE!'
      IERR=100
      CALL CLOSEF(INELM)
      RETURN

  700 FORMAT(' LM=',I5,'  NOD=',I6,' APPEARS ',I2,' TIMES'/)
20001 FORMAT(' ERROR: PERMEABILITY CLASS ',I4,' OF ELEMENT ',I5,
     &       ' NOT GIVEN!')
C----------END LECELM-----------
      END


      SUBROUTINE LECPE(IERR)

      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------

      DIMENSION PERM(6)

C     INPUT OF THE PERMEABILITIES [K]=(K11,K12,K22,K13,K23,K33)
C     ---------------------------------------------------------

C      WRITE(OUTRA,700)

      CALL FINDKW(INPAR,'PERMEABILITIES',NBLINE)

      DO 10 I=1,6
   10 PERM(I)=0D0

      IERR=0
  100 READ(INPAR,*,IOSTAT=IOS) NP,PORO,PERM
      IF (IOS.NE.0.OR.NP.LT.1) GOTO 600
      IF (NP.GT.IV3) GOTO 500
        PORV(NP)=PORO
        IF(.NOT.(PERM(3).GT.0D0.OR. PERM(6).GT.0D0)) THEN
          PERM(3)=PERM(1)
          PERM(6)=PERM(1)
C           WRITE(OUTRA,720)NP,PORO,PERM(1)
C         ELSE
C           WRITE(OUTRA,730)NP,PORO,PERM(1),PERM(2),PERM(4),
C     &                       PERM(3),PERM(5),PERM(6)
```

```
        ENDIF
        DO 200 I=1,6
          PERA(I,NP)=PERM(I)
          PERM(I)=0D0
  200   CONTINUE
      GOTO 100
  500 WRITE (OUTRA,710)
      IERR=10
  600 CONTINUE
      CALL CLOSEF(INPAR)
      RETURN
  700 FORMAT (/5X,'VALUES  OF PERM/PORO CLASSES    '/5X,27('*')/)
  710 FORMAT (/' ERROR IN LECPE.CHECK DIMENSION OF PERM.CLASSES')
  720 FORMAT(5X,I5,5X,1P,2E14.5)
  730 FORMAT(5X,I5,5X,1P,4E14.5/43X,2E14.5/57X,E14.5)
C----------END LECPE-----------
      END


      SUBROUTINE CONMAT(NEWCON,FICON,ICON2,NBLINE,IERR)
C ---------------------------------------------------------------------
C Calculate connection matrix
C ---------------------------------------------------------------------

      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
      LOGICAL KRTEST,NEWCON,ICON2,A
      DIMENSION ICONN(6)
      CHARACTER*(*) FICON

C Initialize
      DO 10 I=1,6
      DO 10 J=1,MXLM
10    ICON(I,J)=0
      IERR=0
      MXL1D=0
      MXL2D=0
      MXL3D=0

      IF(.NOT.NEWCON.AND..NOT.ICON2) GOTO 400
      IF(.NOT.NEWCON.AND.ICON2) GOTO 500
C Mark pinched nodes with negativ signs
      DO 15 J=1,MXLM
         IF(NARV(J).LT.0) THEN
            DO 12 I1=1,KRV(J)
               N1=ABS(NELC(I1,J))
               DO 12 I2=I1+1,KRV(J)
                  IF(N1.EQ.ABS(NELC(I2,J))) THEN
                     NELC(I1,J)=-N1
                     NELC(I2,J)=-N1
                  ENDIF
12          CONTINUE
         ENDIF
15    CONTINUE

C Check element types, store 1D-elements into IELV
```

```
      DO 20 LM=1,MXLM
         IF(.NOT.KRTEST(KRV(LM))) THEN
            WRITE(OUTRA,20000) IELM(LM),KRV(LM)
            IERR=500
            GOTO 20
         ENDIF
         IF(KRV(LM).EQ.3) THEN
            MXL1D=MXL1D+1
            IELV(MXL1D)=LM
         ENDIF
20    CONTINUE

C Add 2D-elements in IELV
      DO 30 LM=1,MXLM
         IF(KRV(LM).GE.6 .AND. KRV(LM).LE.9) THEN
            MXL2D=MXL2D+1
            IELV(MXL1D+MXL2D)=LM
         ENDIF
30    CONTINUE

C Add 3D-elements in IELV
      DO 40 LM=1,MXLM
         IF(KRV(LM).GE.10) THEN
            MXL3D=MXL3D+1
            IELV(MXL1D+MXL2D+MXL3D)=LM
         ENDIF
40    CONTINUE

C Connection of 1D-elements
      DO 100 LMD=1,MXL1D
         CALL FACE1D(IELV,LMD,MXL1D,MXL2D,MXL3D,MXLM,KRV,ICON,NELC)
100   CONTINUE

C Connection of 2D-elements
      DO 200 LMD=MXL1D+1,MXL1D+MXL2D
        CALL FACE2D(IELV,LMD,MXL1D,MXL2D,MXL3D,MXLM,KRV,ICON,NELC)
200   CONTINUE

C Connection of 3D-elements
      DO 300 LMD=MXL1D+MXL2D+1,MXLM
        CALL FACE3D(IELV,LMD,MXL1D,MXL2D,MXL3D,MXLM,KRV,ICON,NELC)
300   CONTINUE

C Reset sign of nodes in pinched elements
      DO 315 J=1,MXLM
         IF(NARV(J).LT.0) THEN
            DO 312 I=1,KRV(J)
312         NELC(I,J)=ABS(NELC(I,J))
         ENDIF
315   CONTINUE

      CALL SAVCONM(NEWCON,ICON2,FICON)
      CALL CLOSEF(IOMAT)
      RETURN

C -- Read CONNECTION MATRIX in old format and write new format version
400   DO 410 J=1,MXLM
410   READ(IOMAT,*,ERR=620,END=630) IDUM,IDUM,(ICON(I,J),I=1,6)
      A=.TRUE.
C      CALL QUEST(' UPDATE CONNECTION MATRIX [Y/N,def=N] ',A)
      PRINT *,'CONNECTION MATRIX is being written in new format'
      IF (A)  CALL SAVCONM(NEWCON,ICON2,FICON)
      CALL CLOSEF(IOMAT)
      RETURN

C -- Read CONNECTION MATRIX in new format
500   DO 510 J=1,MXLM
         READ(IOMAT,*,ERR=620,END=630) IEL,ICONN
         IF ((IEL.LT.1.OR.IEL.GT.IV2).OR.
     &        (INVELM(IEL).LT.1.OR.INVELM(IEL).GT.MXLM)) GOTO 620
         IEL=INVELM(IEL)
         DO 505 I=1,6
            IF (ICONN(I).GT.0) THEN
               ICON(I,IEL)=INVELM(ICONN(I))
            ELSE
```

```
              ICON(I,IEL)=ICONN(I)
            ENDIF
505       CONTINUE
510    CONTINUE
       CALL CLOSEF(IOMAT)
C --- Simple test if ICON is complete
       DO 520 J=1,MXLM
520    IF (ICON(1,J).EQ.0) GOTO 650
       RETURN

620    WRITE(OUTRA,'(A,I5)') 'ERROR IN CONNECTION MATRIX FILE AT LINE ',
     &                        NBLINE+J
       IERR=1
       CALL CLOSEF(IOMAT)
       RETURN
630    WRITE(OUTRA,'(A)') 'EOF IN CONNECTION MATRIX FILE'
       IERR=2
       CALL CLOSEF(IOMAT)
       RETURN
650    WRITE(OUTRA,'(A)') 'CONNECTION MATRIX FILE IS INCOMPLETE'
       IERR=3
       RETURN
20000  FORMAT(/' ERROR: ILLEGAL NUMBER OF NODES FOR ELEMENT ',I5,
     &         ' : KR =',I5)
C----------END CONMAT-----------
       END



       SUBROUTINE SAVCONM(NEWCON,ICON2,FICON)
       IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON BLANK_COM-------

       PARAMETER (IV1=100000,IV2=10000,IV3=1000)

       COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
      &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
      &          ,XORBLK(3)
      &          ,NELP(IV2),KRV(IV2),NARV(IV2)
      &          ,PERA(6,IV3),PORV(IV3)
      &          ,MXLM,MNNIC,MXNIC,MAXDIM
      &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
       INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
       COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
      &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
       LOGICAL NEWCON,ICON2
       CHARACTER FICON2*60,FICON*(*),EXT*5
       DIMENSION ICONN(7)

C  get new file name if NEWCON is false and old ICON was read
       FICON2=FICON
       IF (.NOT.NEWCON.AND..NOT.ICON2) THEN
          CALL BLKOUT(FICON2,LF)
          IF (LF.GT.3) THEN
             EXT=FICON2(LF-3:LF)//' '
             CALL UPCAS2(EXT)
          ELSE
             EXT=' '
          ENDIF
          IF (EXT.EQ.'_CON ') THEN
             FICON2(LF-3:LF+1)='_CON2'
          ELSE
             FICON2(LF+1:LF+4)='_CON2'
          ENDIF
          CALL CLOSEF(IOMAT)
          CALL NEWDEFL(IOMAT,FICON2)
       ENDIF
       REWIND (IOMAT)
       WRITE(IOMAT,'(A,I5)') 'Number of elements with PERM>0 :',MXLM
       WRITE(IOMAT,'(/A)')   'CONNECT2'
       DO 100 J=1,MXLM
```

```
          ICONN(1)=IELM(J)
          DO 50 I=1,6
             IF (ICON(I,J).GT.0) THEN
                ICONN(I+1)=IELM(ICON(I,J))
             ELSE
                ICONN(I+1)=ICON(I,J)
             ENDIF
50        CONTINUE
          WRITE(IOMAT,'(716)') ICONN
100    CONTINUE
       ENDFILE(IOMAT)
       RETURN
C----------END SAVCONM----------
       END


       SUBROUTINE FACE1D(IELV,LMD,MXL1D,MXL2D,MXL3D,MXLM,KRV,ICON,NELC)

       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION IELV(*),KRV(*),ICON(6,*),NELC(27,*)
       DIMENSION ITYPES(8),NFACES(3),NNODES(4,3)

       DATA ITYPES /0,0,1,0,0,2,0,3/
       DATA NFACES /4,3,4/
       DATA NNODES / 1, 3, 2, 2,  !* line element
      &              2, 4, 6, 0,  !* triangle
      &              2, 4, 6, 8/  !* rectangle

C Get parameters for this element type
C  KR=3  --> TYPE 1  (line)
C  KR=6  --> TYPE 2  (triangle)
C  KR=8  --> TYPE 3  (rectangle)

       LM=IELV(LMD)

C Loop over all faces of element LM
C ---------------------------------
       DO 300 NF=1,4
          IF(ICON(NF,LM) .NE. 0) GOTO 300

C Variables for this face
          IF(NF.LE.2) THEN
             LMST=LMD+1
          ELSEIF (NF.EQ.3) THEN
             LMST=MXL1D+1
          ELSE
             LMST=NNFOUND+1
          ENDIF

C Get node to be compared
          NOD1=NELC(NNODES(NF,1),LM)

C Loop over all possible neighboured elements not yet treated
          DO 200 NN=LMST,MXLM
             NNFOUND=NN
             LMN=IELV(NN)
             KRN=KRV(LMN)
             IF(((NF.LE.2).AND.(KRN.EQ.3)).OR.
      &         ((NF.GT.2).AND.(NDIM(KRN).EQ.2))) THEN
                ITYPEN=ITYPES(KRN)
                IF(ITYPEN.LT.1) GOTO 200
                NFACEN=NFACES(ITYPEN)
                DO 100 NNF=1,NFACEN
                   IF(NOD1.NE.ABS(NELC(NNODES(NNF,ITYPEN),LMN))) GOTO 100
                   ICON(NF,LM)=LMN
                   ICON(NNF,LMN)=LM
                   GOTO 300
100             CONTINUE
             ELSE
                DO 150 I=1,KRN
                   IF(NOD1.NE.ABS(NELC(I,LMN))) GOTO 150
                   ICON(NF,LM)=LMN
                   GOTO 300
150             CONTINUE
```

```
            ENDIF
200       CONTINUE

C No neighboured face found
      ICON(NF,LM)=-1

300   CONTINUE
      RETURN
C----------END FACE1D----------
      END


      SUBROUTINE FACE2D(IELV,LMD,MXL1D,MXL2D,MXL3D,MXLM,KRV,ICON,NELC)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION IELV(*),KRV(*),ICON(6,*),NELC(27,*)
      DIMENSION ITYPES(27),NFACES(6),NNODES(4,6,6),NOD(4)

      DATA ITYPES /5*0,1,0,2,0,3,0,0,4,0,5,4*0,6,7*0/
      DATA NFACES /5,6,4,5,5,6/
      DATA ((NNODES (I,J,1),I=1,4),J=1,6)
     &             / 2, 0, 0, 0,  !* triangle face 1
     &               4, 0, 0, 0,  !* triangle face 2
     &               6, 0, 0, 0,  !* triangle face 3
     &               2, 4, 6, 0,  !* triangle face 4
     &               2, 4, 6, 0,  !* triangle face 5
     &               0, 0, 0, 0/  !* dummy face 6

      DATA ((NNODES (I,J,2),I=1,4),J=1,6)
     &             / 2, 0, 0, 0,  !* rectangle face 1
     &               4, 0, 0, 0,  !* rectangle face 2
     &               6, 0, 0, 0,  !* rectangle face 3
     &               8, 0, 0, 0,  !* rectangle face 4
     &               2, 4, 6, 8,  !* rectangle face 5
     &               2, 4, 6, 8/  !* rectangle face 6

      DATA ((NNODES (I,J,3),I=1,4),J=1,6)
     &             / 2, 4, 6, 0,  !* tetra face 1
     &               2, 7, 8, 0,  !* tetra face 2
     &               4, 8, 9, 0,  !* tetra face 3
     &               6, 9, 7, 0,  !* tetra face 4
     &               0, 0, 0, 0,  !* dummy face 5
     &               0, 0, 0, 0/  !* dummy face 6

      DATA ((NNODES (I,J,4),I=1,4),J=1,6)
     &             / 2, 4, 6, 8,  !* pyram face 1
     &               2, 9,10, 0,  !* pyram face 2
     &               4,10,11, 0,  !* pyram face 3
     &               6,11,12, 0,  !* pyram face 4
     &               8,12, 9, 0,  !* pyram face 5
     &               0, 0, 0, 0/  !* dummy face 6

      DATA ((NNODES (I,J,5),I=1,4),J=1,6)
     &             / 2, 4, 6, 0,  !* prism face 1
     &              11,13,15, 0,  !* prism face 2
     &               2, 8,11, 7,  !* prism face 3
     &               4, 9,13, 8,  !* prism face 4
     &               6, 7,15, 9,  !* prism face 5
     &               0, 0, 0, 0/  !* dummy face 6

      DATA ((NNODES (I,J,6),I=1,4),J=1,6)
     &             / 4,10,16,11,  !* brick face 1
     &               8,12,20, 9,  !* brick face 2
     &               6,11,18,12,  !* brick face 3
     &               2, 9,14,10,  !* brick face 4
     &              14,16,18,20,  !* brick face 5
     &               2, 4, 6, 8/  !* brick face 6

C Get parameters for this element type
C  KR=6  --> TYPE 1  (triangle)
C  KR=8  --> TYPE 2  (rectangle)
C  KR=10 --> TYPE 3  (tetrahedron)
C  KR=13 --> TYPE 4  (pyramid)
C  KR=15 --> TYPE 5  (prisma)
```

```
C  KR=20 --> TYPE 6  (brick)

      LM=IELV(LMD)
      KR=KRV(LM)
      ITYPE=ITYPES(KR)
      IF(ITYPE.LT.1) RETURN
      NFACE=NFACES(ITYPE)

C Loop over all faces of element LM
C ---------------------------------
      DO 300 NF=1,NFACE
         IF(ICON(NF,LM) .NE. 0) GOTO 300

C Variables for this face
         IF(NF.LE.NFACE-2) THEN
            N3MAX=1
            LMST=LMD+1
         ELSEIF (NF.EQ.NFACE-1) THEN
            N3MAX=3
            LMST=MXL1D+MXL2D+1
         ELSE
            N3MAX=3
            LMST=NNFOUND+1
         ENDIF

C Get N3MAX nodes for face NF (only non-repetitive nodes!)
         N3=0
         DO 10 I=1,4
            NOD1=NELC(NNODES(I,NF,ITYPE),LM)
            IF(NOD1.GT.0) THEN
               N3=N3+1
               NOD(N3)=NOD1
               IF(N3.GE.N3MAX) GOTO 20
            ENDIF
10       CONTINUE

         ICON(NF,LM)=-2
         GOTO 300

C Loop over all possible neighboured 3D-elements not yet treated
20       DO 200 NN=LMST,MXLM
            NNFOUND=NN
            LMN=IELV(NN)
            KRN=KRV(LMN)
            ITYPEN=ITYPES(KRN)
            IF(ITYPEN.LT.1) GOTO 200
            NFACEN=NFACES(ITYPEN)
            DO 100 NNF=1,NFACEN
               IF (NNODES(2,NNF,ITYPEN).GT.0) THEN
                 IF(NNODES(4,NNF,ITYPEN).GT.0) THEN
                   NMID=4
                 ELSE
                   NMID=3
                 ENDIF
               ELSE
                 NMID=1
               ENDIF
               NFOUND=0
               DO 50 K=1,NMID
                  KK=NNODES(K,NNF,ITYPEN)
                  DO 40 I=1,N3MAX
                     IF(NOD(I).EQ.ABS(NELC(KK,LMN))) NFOUND=NFOUND+1
40                CONTINUE
                  IF(NFOUND.GE.N3MAX) THEN
                     ICON(NF,LM)=LMN                              !* found!
                     IF((NF.LE.NFACE-2 .AND. KRN.LT.10) .OR.
     &                   NF.GT.NFACE-2) ICON(NNF,LMN)=LM
                     GOTO 300
                  ENDIF
50             CONTINUE
100      CONTINUE
200   CONTINUE

C No neighboured face found
```

```
250   ICON(NF,LM)=-1
300   CONTINUE
      RETURN
C----------END FACE2D-----------
      END


      SUBROUTINE FACE3D(IELV,LMD,MXL1D,MXL2D,MXL3D,MXLM,KRV,ICON,NELC)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION IELV(*),KRV(*),ICON(6,*),NELC(27,*)
      DIMENSION ITYPES(27),NFACES(4),NNODES(4,6,4),NOD(4)

      DATA ITYPES /9*0,1,0,0,2,0,3,4*0,4,7*0/
      DATA NFACES /4,5,5,6/
      DATA ((NNODES (I,J,1),I=1,4),J=1,6)
     &             / 2, 4, 6, 0,   !* tetra face 1
     &               2, 7, 8, 0,   !* tetra face 2
     &               4, 8, 9, 0,   !* tetra face 3
     &               6, 9, 7, 0,   !* tetra face 4
     &               0, 0, 0, 0,   !* dummy face 5
     &               0, 0, 0, 0/   !* dummy face 6

      DATA ((NNODES (I,J,2),I=1,4),J=1,6)
     &             / 2, 4, 6, 8,   !* pyram face 1
     &               2, 9,10, 0,   !* pyram fac2
     &               4,10,11, 0,   !* pyram face 3
     &               6,11,12, 0,   !* pyram face 4
     &               8,12, 9, 0,   !* pyram face 5
     &               0, 0, 0, 0/   !* dummy face 6

      DATA ((NNODES (I,J,3),I=1,4),J=1,6)
     &             / 2, 4, 6, 0,   !* prism face 1
     &              11,13,15, 0,   !* prism face 2
     &               2, 8,11, 7,   !* prism face 3
     &               4, 9,13, 8,   !* prism face 4
     &               6, 7,15, 9,   !* prism face 5
     &               0, 0, 0, 0/   !* dummy face 6

      DATA ((NNODES (I,J,4),I=1,4),J=1,6)
     &             / 4,10,16,11,   !* brick face 1
     &               8,12,20, 9,   !* brick face 2
     &               6,11,18,12,   !* brick face 3
     &               2, 9,14,10,   !* brick face 4
     &              14,16,18,20,   !* brick face 5
     &               2, 4, 6, 8/   !* brick face 6

C Get parameters for this element type
C  KR=10 --> TYPE 1  (tetrahedron)
C  KR=13 --> TYPE 2  (pyramid)
C  KR=15 --> TYPE 3  (prisma)
C  KR=20 --> TYPE 4  (brick)

      LM=IELV(LMD)
      KR=KRV(LM)
      ITYPE=ITYPES(KR)
      IF(ITYPE.LT.1) RETURN
      NFACE=NFACES(ITYPE)

C Loop over all faces of element LM
C ---------------------------------
      DO 300 NF=1,NFACE
         IF(ICON(NF,LM) .NE. 0) GOTO 300

C Get 3 nodes for face NF (only non-repetitive nodes!)
         N3=0
         IF=4
         IF (NNODES(4,NF,ITYPE).EQ.0) IF=3
         DO 30 I=1,IF
            NOD1=NELC(NNODES(I,NF,ITYPE),LM)
            IF(NOD1.GT.0) THEN
               N3=N3+1
               NOD(N3)=NOD1
            ENDIF
```

```
30          CONTINUE
            IF(N3.LT.3) THEN
               ICON(NF,LM)=-2
               GOTO 300
            ENDIF

C Loop over neighboured 3D-elements
            DO 200 J=LMD+1,MXLM
               LMN=IELV(J)
               KRN=KRV(LMN)
               ITYPEN=ITYPES(KRN)
               IF(ITYPEN.LT.1) GOTO 200
               NFACEN=NFACES(ITYPEN)
               DO 100 NNF=1,NFACEN
                  NMID=3
                  IF (NNODES(4,NNF,ITYPEN).GT.0) NMID=4
                  NFOUND=0
                  DO 50 K=1,NMID
                     KK=NNODES(K,NNF,ITYPEN)
                     DO 40 I=1,3
                        IF(NOD(I).EQ.ABS(NELC(KK,LMN))) NFOUND=NFOUND+1
40                   CONTINUE
                     IF(NFOUND.GT.2) THEN
                        ICON(NF,LM)=LMN    !* found!
                        ICON(NNF,LMN)=LM
                        GOTO 300
                     ENDIF
50                CONTINUE
100         CONTINUE
200      CONTINUE

C No neighboured face found
250      ICON(NF,LM)=-1
300      CONTINUE
         RETURN
C-----------END FACE3D-----------
         END



         SUBROUTINE STREAM(USER)
C---------------------- STREAM -----------------------------------------
C               GUIDING THE PARTICLE TRACKING
C-----------------------------------------------------------------------

         IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON START_COM--------
         INTEGER DIMSTA
         LOGICAL RDNODE,KWNODE
         COMMON /START/ NTRAJ,MAXSTEP,NRINTR,IELSTA,XGSTART(3),XLSTART(3)
        &               ,DIMSTA,MXTRAJ
        &               ,RDNODE,KWNODE,MXNODE,NBNODE
C----------END START_COM--------
C-----COMMON TAPES_COM-------
         INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
         COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
        &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------

         CHARACTER USER*(*),TEXT*17,MESSAGE*132
         CALL RDSTART(*1000)
100      CONTINUE
         CALL FINDSTA(*1000,*100)
         CALL INITTRA(*120,MESSAGE)
         CALL TRAJECT(MESSAGE)
120      WRITE(TEXT,'(A,I5,A)') ' TRACK NO.',NRINTR,': '
         NB=INDEX(MESSAGE,':')
         NB=NB-1
         IF(NB.LT.1) NB=116
         CALL CPUTIM(OUTRA,TEXT//MESSAGE(1:NB),USER)
         GOTO 100
1000     CONTINUE
         RETURN
C----------END STREAM----------
         END
```

```
      SUBROUTINE RDSTART(*)

      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON START_COM--------
      INTEGER DIMSTA
      LOGICAL RDNODE,KWNODE
      COMMON /START/ NTRAJ,MAXSTEP,NRINTR,IELSTA,XGSTART(3),XLSTART(3)
     &              ,DIMSTA,MXTRAJ
     &              ,RDNODE,KWNODE,MXNODE,NBNODE
C----------END START_COM---------
C-----COMMON STEP_COM-----------

      PARAMETER (MAXITER=100,DFSTEP=1D-1,SUBRA=5D-4
     &          ,SUBSTEP=2D0,NCHECK=10,DFDSCR=1D-1
     &          ,DEFDIST=1D-2)

      LOGICAL   BNDMAX,KUTTA
      COMMON /STEP/ NITTOT,TIMETOT,DISTTOT,NITELM,TIMEELM,DISTELM,DISTO
     &             ,XGSTEP(3),DXSTEP,DSSTEP,DISCRQ,NCHK,BNDMAX,KUTTA
     &             ,DISTMNQ,XGODIST(3)
C----------END STEP_COM---------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM---------
      COMMON /ORIGIN/ XORIG(3)


C-----COMMON ARRIVE_COM------
      PARAMETER (DEFRARR=1D0)
      LOGICAL ATDSTR,ARRIVED
      CHARACTER FIATD*256
      COMMON /ARRIVE/ ATDSTR,XARRIVE(3),RARRIVE
     &     ,TARRIVE,DARRIVE,IARRIVE,FIATD,ARRIVED
C----------END ARRIVE_COM-------

C  -- Input of arrival time distribution parameters
      CALL FINDKW(INSTA,'@ARRIVAL',NB)
      ATDSTR=(NB.GT.0)
      IF (ATDSTR) THEN
C       CALL RDFNAME(INSTA,FIATD)
C       IF (FIATD(:1).EQ.' ') GOTO 100
        READ (INSTA,'(A)',ERR=100) FIATD
        READ (INSTA,*,ERR=100) XARRIVE,RARRIVE
        IF (RARRIVE.LE.0) RARRIVE=DEFRARR
        CALL BLKOUT(FIATD,LFI)
        CALL NEWDEFL(OUATD,FIATD(:LFI))
        WRITE (OUATD,1500) XARRIVE,RARRIVE
        RARRIVE=RARRIVE**2
        CALL VDIF(3,XARRIVE,XORIG,XARRIVE)
      ENDIF

C  -- Local step size
      CALL FINDKW(INSTA,'@STEP',NB)
      IF (NB.GT.0) THEN
        READ (INSTA,*,ERR=100) DSSTEP
      ELSE
        DSSTEP=DFSTEP
      ENDIF

C  -- Input of small region discretization
      CALL FINDKW(INSTA,'@DISCRETE',NB)
      IF (NB.GT.0) THEN
        READ (INSTA,*,ERR=100) DSCR
      ELSE
        DSCR=DFDSCR
      ENDIF
      DISCRQ=DSCR**2

C  --Input of Boundary threshold selection
      ISWB=0
      CALL FINDKW(INSTA,'@BOUND',NB)
      IF (NB.GT.0) READ (INSTA,*,ERR=100,END=100) ISWB
      BNDMAX=(ISWB.EQ.1)
```

```
C  -- Runge Kutta approximation (Default: KUTTA=.TRUE.)
      ISWK=1
      CALL FINDKW(INSTA,'@KUTTA',NB)
      IF (NB.GT.0) READ (INSTA,*,ERR=100,END=100) ISWK
      KUTTA=(ISWK.EQ.1)


C  -- Input of mininum distance of path coordinates (def: DEFDIST)
      DISTMIN=DEFDIST
      CALL FINDKW(INSTA,'@OUTPUT DISTANCE',NB)
      IF (NB.GT.0) READ (INSTA,*,ERR=100,END=100) DISTMIN
      DISTMNQ=SIGN(DISTMIN**2,DISTMIN)

C  --Input of keyword for nodal starting point values
      KWNODE=.FALSE.
      RDNODE=.FALSE.
      CALL FINDKW(INSTA,'@NODE',NB)
      IF (NB.GT.0) THEN
         READ(INSTA,*,ERR=100,END=100) MXNODE
         KWNODE=(MXNODE.GT.0)
      ENDIF
      MXNODE=MAX(0,MXNODE)

C  --Input of starting points
      CALL FINDKW(INSTA,'START',NB)
      IF (NB.LT.1) GOTO 100
      READ (INSTA,*,ERR=100,END=100) NTRAJ,MAXSTEP
      MXTRAJ=NTRAJ+MXNODE
      CALL CNTWORD(INSTA,ND,IEND)
      DIMSTA=MIN(ND-1,3)
      WRITE(OUTRA,900)
      WRITE (OUTRA,1000) MXTRAJ,MAXSTEP,MXNODE,NTRAJ
      WRITE (OUTRA,1100) DSSTEP,DSCR,DISTMIN,ISWB,ISWK
      RETURN
  100 WRITE (OUTRA,2000)
      RETURN 1
  900 FORMAT(///4X,' RESULTS OF THE TRACKING '/
     &       4X,' ********************** ')
 1000 FORMAT (//3X,'NUMBER OF TRAJECTORIES..... ..: ',I5/
     &          3X,'MAXIMUM NUMBER OF STEPS PER TRAJECTORY...: ',I5/
     &          3X,'NUMBER OF STARTING NODES.....: ',I5/
     &          3X,'NUMBER OF STARTING POINTS.....: ',I5)
 1100 FORMAT (//3X,'LOCAL STEP SIZE .....: ',F6.4/
     &          3X,'SUBDISCRETE REGION DISTANCE.....: '
     &            ,F6.4,' [m]',/
     &          3X,'MINIMUM DISTANCE OF TRAJ. COORDINATES....: '
     &            ,F6.4,' [m]',/
     &          3X,'BOUNDARY THRESHOLD SELECTION.....: ',I2/
     &          3X,'  (0 = ENTER FLUX TAKING 3D-ELEMENTS,'/
     &          3X,'   1 = FOLLOW HIGHEST BOUNDARY VELOCITY)'/
     &          3X,'RUNGE KUTTA INTEGRATION.....: ',I2/
     &          3X,'  (0 = OFF,'/
     &          3X,'   1 = ON )')
 1500 FORMAT (//4X,'ARRIVAL TIME DISTRIBUTION'
     &         /4X,'************************'
     &        //5X,'Final Point :',1P,3E15.6,' [m]'
     &        //5X,'Radius      :',1P,E15.6,' [m]'
     & ///' Path   ','Arrv[1/0]',' X Start [m]   ',' Y Start [m]
     &  ,' Z Start [m]   ',' Time [y]    ','Distance [m]   ')
 2000 FORMAT ('--ERROR IN READING STARTING POINT FILE--')
C----------END RDSTART----------
      END


      SUBROUTINE FINDSTA(*,*)

      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
```

```
     &           ,XORBLK(3)
     &           ,NELP(IV2),KRV(IV2),NARV(IV2)
     &           ,PERA(6,IV3),PORV(IV3)
     &           ,MXLM,MNNIC,MXNIC,MAXDIM
     &           ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
C-----COMMON START_COM--------
      INTEGER DIMSTA
      LOGICAL RDNODE,KWNODE
      COMMON /START/ NTRAJ,MAXSTEP,NRINTR,IELSTA,XGSTART(3),XLSTART(3)
     &              ,DIMSTA,MXTRAJ
     &              ,RDNODE,KWNODE,MXNODE,NBNODE
C----------END START_COM--------
      COMMON /ORIGIN/ XORIG(3)



      DIMENSION XX(3)
      DATA NTR /0/

      NTR=NTR+1
      IF (NTR.GT.MXTRAJ) RETURN 1

C Process starting nodes after starting coordinates (NTR.GT.NTRAJ)
      IF (KWNODE.AND.NTR.GT.NTRAJ) THEN
C Position file 2 lines after keyword for first node
         IF (.NOT.RDNODE) THEN
           CALL FINDKW(INSTA,'NODE',NBL)
           READ (INSTA,*,ERR=100) NDUM
           RDNODE=.TRUE.
           NBNODE=0
         ENDIF
         NBNODE=NBNODE+1
         IF (NBNODE.GT.MXNODE) RETURN 1
C Read starting node and get local coordinates directly
         READ (INSTA,*,ERR=100,END=100) NRINTR,NODE
         IF (NODE.LT.1.OR.NODE.GT.IV1) GOTO 300
         IF (IDCOR(NODE).LT.1) GOTO 300
         WRITE (OUTRA,1000) NRINTR
         CALL VEQV(3,XGSTART,COOR(1,NODE))
         DO 40 IEL=1,MXLM
            KR=KRV(IEL)
            DO 40 NNODE=1,KR
               IF (NELC(NNODE,IEL).NE.NODE) GOTO 40
               IELSTA=IEL
               CALL XLOCAL(KR,NNODE,XLSTART)
               RETURN
40       CONTINUE
         GOTO 300 !* no element found
      ENDIF

      CALL CNTWORD(INSTA,NBWRD,IEND)
      IELFIRS=0
      IF (NBWRD.LT.5) THEN
        READ (INSTA,*,ERR=100,END=100) NRINTR,(XX(I),I=1,DIMSTA)
      ELSE
        READ (INSTA,*,ERR=100,END=100) NRINTR,(XX(I),I=1,3),IELFIRS
      ENDIF
      DO 50 I=DIMSTA+1,3
   50 XX(I)=0D0
      CALL VDIF(3,XX,XORIG,XGSTART)
      WRITE (OUTRA,1000) NRINTR
      CALL FNDELXL(IELFIRS,*200)
      RETURN
  100 WRITE (OUTRA,1100)
      RETURN 1
  200 WRITE (OUTRA,1200)
      RETURN 2
  300 WRITE (OUTRA,1300)
```

```
      RETURN 2

 1000 FORMAT (//6X,'TRACK NO.',I5,
     &         //'ELEMENT K-CLASS NITTOT    LENGTH [m] ',
     &           '   TIME [y]      VELOCITY [m/y]',
     &           '   CUM LEN [m]    CUM TIME [y]   MEAN VEL [m/y]')
 1100 FORMAT ('--ERROR IN READING STARTING POINT FILE--')
 1200 FORMAT (/'--- ELEMENT AND POINT NOT FOUND----')
 1300 FORMAT (/'--- ILLEGAL NODE NUMBER FOR STARTING POINT---')
C----------END FINDSTA----------
      END


      SUBROUTINE FNDELXL(IELFIRS,*)

      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
C-----COMMON START_COM--------
      INTEGER DIMSTA
      LOGICAL RDNODE,KWNODE
      COMMON /START/ NTRAJ,MAXSTEP,NRINTR,IELSTA,XGSTART(3),XLSTART(3)
     &              ,DIMSTA,MXTRAJ
     &              ,RDNODE,KWNODE,MXNODE,NBNODE
C----------END START_COM--------

      PARAMETER (ERR=1.D-4,ERRQ=ERR**2,EPSG=-1D0)
      LOGICAL INS
      DIMENSION DX(3),NAEX(4),XKR(3,27)

C  --- GET NEAREST NODE TO GIVEN STARTING POINT

      CALL VDIF(3,COOR(1,MNNIC),XGSTART,DX)
      DSQMIN=VSCP(3,DX,DX)
      NICMIN=MNNIC
      DO 100 I=MNNIC+1,MXNIC
        IF (IDCOR(I).GT 0) THEN
          CALL VDIF(3,COOR(1,I),XGSTART,DX)
          DSQ=VSCP(3,DX,DX)
          IF (DSQ.LT.DSQMIN) THEN
            NICMIN=I
            DSQMIN=DSQ
          ENDIF
        ENDIF
  100 CONTINUE

      NEL=0
      IF(IELFIRS.GT.0) THEN
          IELV(1)=INVELM(IELFIRS)
          IF(IELV(1).GT.0 .AND. IELV(1).LE.MXLM) NEL=1
      ENDIF
C  --- GET ELEMENTS WITH NEAREST POINT---

      DO 300 J=1,MXLM
        DO 200 I=1,KRV(J)
          IF (NELC(I,J).EQ.NICMIN) THEN
            NEL=NEL+1
```

```
            IELV(NEL)=J
            GOTO 300
          ENDIF
  200   CONTINUE
  300 CONTINUE

C  --- SEARCHING LOCAL POINT IN THOSE ELEMENTS

      DO 400 I=1,NEL
        IEL=IELV(I)
        KR=KRV(IEL)
        N=NDIM(KR)
        CALL GETXKR(KR,COOR,NELC(1,IEL),XKR)
        CALL LSTART(KR,DX)
        CALL SEARCH(XGSTART,XLSTART,DX,KR,XKR
     &            ,SIGN(MAXDIM,NARV(IEL)),EPSG,IERR)
        IF (IERR.NE.0) GOTO 400
        CALL IOTEST(DX,XLSTART,DX,KR,NAEX,NB,INS,.FALSE.,.TRUE.
     &             ,.FALSE.,.FALSE.,.FALSE.,0,0,IERR)
        IF (INS) THEN
          IELSTA=IEL
          RETURN
        ELSE
          CALL VDIF(N,DX,XLSTART,DX)
          IF (VSCP(N,DX,DX).LT.ERRQ) THEN
            IELSTA=IEL
            RETURN
          ENDIF
        ENDIF
  400 CONTINUE
      RETURN 1
C----------END FNDELXL----------
      END


      SUBROUTINE INITTRA(*,MESSAGE)

      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
C-----COMMON START_COM--------
      INTEGER DIMSTA
      LOGICAL RDNODE,KWNODE
      COMMON /START/ NTRAJ,MAXSTEP,NRINTR,IELSTA,XGSTART(3),XLSTART(3)
     &              ,DIMSTA,MXTRAJ
     &              ,RDNODE,KWNODE,MXNODE,NBNODE
C----------END START_COM--------
C-----COMMON BOUND_COM----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
```

```
     &    ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &    ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &     BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &    ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &    ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &     XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &    ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &    ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &    ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &    ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------
      LOGICAL INS
      CHARACTER*(*) MESSAGE
      DIMENSION X0(3)

      CALL VEQV(3,XGBND,XGSTART)
      IPBND=0
      IELPOS=1
      IELBND(1)=IELSTA
      KRBND(1)=KRV(IELSTA)
      CALL VEQV(NDIM(KRBND(1)),XLBND,XLSTART)
      PORBND(1)= PORV(NELP(IELSTA))
      CALL GETPOT(KRBND(1),NELC(1,IELSTA),POT,HEADBND)
      CALL GETPERM(PERA(1,NELP(IELSTA)),PERMBND)
      CALL GETXKR(KRBND(1),COOR,NELC(1,IELSTA),XKRBND)
      CALL VELOTXT(IELSTA,XLBND,XKRBND,PERMBND
     &    ,PORBND(1),HEADBND,KRBND(1),NDIM(KRBND(1))
     &    ,QLBND,QGBND,FMBND,FMIBND
     &    ,IFAIL,'INITTRA')
      IF (IFAIL.NE.0) GOTO 1000


C--- CHECK IF STARTING POINT LIES ON ELEMENT BOUNDARY

      CALL LSTART(KRBND(1),X0)
      CALL IOTEST(X0,XLBND,XLBND,KRBND(1),NAEXBND,NBOUND(1),INS
     &  ,.TRUE.,.TRUE.,.TRUE.,.TRUE.,.FALSE.,0,0,IOERR)
      BNDDIM=MAX(NDIM(KRBND(1))-NBOUND(1),0)
      IF (BNDDIM.LT.MAXDIM) THEN  !* Starting point on boundary: get neighbours
        IF (BNDDIM.EQ.MAXDIM-1) THEN
          CALL CONELM             !* 2D element face
        ELSE
          CALL ALLELM(*1100)      !* element edge or corner
        ENDIF
        CALL NXTCOOR(*1200,IELFAIL,IDFAIL)
        CALL NXTVELO(IELFAIL,IDFAIL)
        CALL VELOTST(*1400)
        CALL MAXVELO
      ELSE
        BNDFLOW(1)=.FALSE.
        FLOWDIM=(NDIM(KRV(IELSTA)))
        NEWDIM=.TRUE.
        LOWDIM=(FLOWDIM.LT.MAXDIM)
        CHKBND=.FALSE.
        ONESTEP=.FALSE.
      ENDIF
      RETURN

 1000 WRITE (OUTRA,10000) IELBND(1),KRBND(1),(XLBND(I,1),I=1,3)
      WRITE (OUTAB,19001) NRINTR,0.,0.,0.,'Velocities of starting '//
     &                    'point not found'
      RETURN 1
 1100 WRITE (OUTRA,11000)
      WRITE (OUTAB,19001) NRINTR,0.,0.,0.,'Too many elements '//
     &                    'for COMMON BOUND'
      RETURN 1
 1200 WRITE (OUTRA,12000) IELM(IELFAIL),KRV(IELFAIL)
     &       ,(XLBND(I,IDFAIL),I=1,3)
      WRITE (OUTAB,19001) NRINTR,0.,0.,0.,'Local coordinates '//
     &         'of starting point not found'
```

```
      RETURN 1
 1300 WRITE (OUTRA,13000) IELM(IELFAIL),KRV(IELFAIL)
     &      ,(XLBND(I,IDFAIL),I=1,3)
      WRITE (OUTAB,19001) NRINTR,0.,0.,0.,'Velocities of starting '//
     &                    'point not found'
      RETURN 1
 1400 WRITE (OUTRA,14000)
      WRITE (OUTAB,19001) NRINTR,0.,0.,0.,'Starting point lies in '//
     &                'a corner node'
      RETURN 1
10000 FORMAT (//'  Failure during INITTRA (VELOX) .'/
     &          '   Velocities of starting point not found.'/
     &          '   Element No      : ',I7/
     &          '   KR (El. Type)   : ',I7/
     &          '   at local point : ',1P,3E15.5)
11000 FORMAT (//'  Failure during INITTRA.'/
     &          '   Too many elements for COMMON BOUND')
12000 FORMAT (//'  Failure during INITTRA (NXTCOOR) .'/
     &          '   local coordinates of staring point not found:'/
     &          '   Element No      : ',I7/
     &          '   KR (El Type)    : ',I7/
     &          '   at local point : ',1P,3E15.5)
13000 FORMAT (//'  Failure during INITTRA (NXTVELO) .'/
     &          '   Velocities of starting point not found.'/
     &          '   Element No      : ',I7/
     &          '   KR (El. Type)   : ',I7/
     &          '   at local point : ',1P,3E15.5)
14000 FORMAT (//'  Starting point lies in a corner node.'/
     &          '  Flux points totally outside the elements')
19001 FORMAT (I5,1P,3E15.5,5X,A)
C----------END INITTRA----------
      END


      SUBROUTINE TRAJECT(MESSAGE)

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL NEWELM,WROUT
      CHARACTER*132 MESSAGE
      DIMENSION XGFAIL(3),XG(3)

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
C-----COMMON START_COM--------
      INTEGER DIMSTA
      LOGICAL RDNODE,KWNODE
      COMMON /START/ NTRAJ,MAXSTEP,NRINTR,IELSTA,XGSTART(3),XLSTART(3)
     &              ,DIMSTA,MXTRAJ
     &              ,RDNODE,KWNODE,MXNODE,NBNODE
C----------END START_COM--------
C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM
```

```
      COMMON /IBOUND/
     &     FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &    ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &    ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &     BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &    ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &    ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &     XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &    ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &    ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &    ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &    ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------
C-----COMMON STEP_COM----------

      PARAMETER (MAXITER=100,DFSTEP=1D-1,SUBRA=5D-4
     &           ,SUBSTEP=2D0,NCHECK=10,DFDSCR=1D-1
     &           ,DEFDIST=1D-2)

      LOGICAL   BNDMAX,KUTTA
      COMMON /STEP/ NITTOT,TIMETOT,DISTTOT,NITELM,TIMEELM,DISTELM,DISTO
     &              ,XGSTEP(3),DXSTEP,DSSTEP,DISCRQ,NCHK,BNDMAX,KUTTA
     &              ,DISTMNQ,XGODIST(3)
C----------END STEP_COM---------
      COMMON /ORIGIN/ XORIG(3)


C-----COMMON ARRIVE_COM------
      PARAMETER (DEFRARR=1D0)
      LOGICAL ATDSTR,ARRIVED
      CHARACTER FIATD*256
      COMMON /ARRIVE/ ATDSTR,XARRIVE(3),RARRIVE
     &     ,TARRIVE,DARRIVE,IARRIVE,FIATD,ARRIVED
C----------END ARRIVE_COM-------

C --------- STEPPING THROUGH ELEMENTS (MAX. NO. = MXELMTR)---

      NEWELM=.TRUE.
      TIMEELM=0D0
      DISTELM=0D0
      NITTOT=0
      TIMETOT=0D0
      DISTTOT=0D0
      IELOLD=0
      CALL VEQV(3,XGSTEP,XGBND)
      WRITE (OUPAT,'(''PATH'',I6)') NRINTR
      WRITE (OULVT,'(''PATH'',I6)') NRINTR

C Initialize arrival time distr. parameters
      IF (ATDSTR) THEN
        ARRIVED=.FALSE.
      ENDIF

  500 CONTINUE
      ID=ICYCLE(IPBND,IELPOS,MXBOUND)
      NEWELM=(IELOLD.NE.IELBND(ID))
      IF((NEWELM.OR.NEWDIM) .AND. IELOLD.GT.0)
     &    CALL WRELEM(IELM(IELOLD),NELP(IELOLD))
      IF(NEWELM) IELOLD=IELBND(ID)

      CALL STEPELM(IFAIL,MESSAGE)
      IF (IFAIL.NE.0) GOTO 900

      CALL NEXTELM(*900,MESSAGE)
      GOTO 500

C   --- TRACK LEFT REGION OR STOPPED ---

  900 CONTINUE
```

```
      CALL WRELEM(IELM(IELOLD),NELP(IELOLD))
      VELAVRT=0D0
      IF (TIMETOT.GT.0D0) VELAVRT=DISTTOT/TIMETOT
      WRITE (OUTRA,19000) DISTTOT,TIMETOT,VELAVRT,MESSAGE
      WRITE (OUTAB,19001) NRINTR,DISTTOT,TIMETOT,VELAVRT,MESSAGE
C --Writing arrival time output
      IF (ATDSTR) THEN
        IF (.NOT.ARRIVED) THEN
          IARRIVE=0
          TARRIVE=TIMETOT
          DARRIVE=DISTTOT
        ENDIF
        WRITE(OUATD,20000) NRINTR,IARRIVE,(XGSTART(I)+XORIG(I),I=1,3)
     &    ,TARRIVE,DARRIVE
      ENDIF

      RETURN

C     --- FORMATS ---

19000 FORMAT (/13X,'TOTAL : ',1P,45X,3E15.5//A)
19001 FORMAT (I5,1P,3E15.5,5X,A)
20000 FORMAT (I4,8X,I1,4X,1P,5(E12.5,3X))

C----------END TRAJECT----------
      END


      SUBROUTINE STEPELM(IFAIL,MESSAGE)
C-------------------------------------------------------------
C     STEPPING THROUGH ONE ELEMENT
C
C     ERROR PARAMETER IFAIL
C     IFAIL =  0   : OKAY
C           = 10   : PARAMETER MAXITER EXHAUSTED
C           = 20   : FLUX IS ZERO
C           = 30   : ERROR IN SUBR. RUNGE
C           = 40   : ERROR IN SUBR. VELOX
C           = 50   : TRACK CONCENTRATES WITHIN SMALL REGION
C           = 60   : TRACK OSCILLATING
C-------------------------------------------------------------

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (YEAR=3.15576D7)

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
```

```
     &    ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &    ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &     XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &    ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &    ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &    ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &    ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------
C-----COMMON STEP_COM-----------

      PARAMETER (MAXITER=100,DFSTEP=1D-1,SUBRA=5D-4
     &           ,SUBSTEP=2D0,NCHECK=10,DFDSCR=1D-1
     &           ,DEFDIST=1D-2)

      LOGICAL   BNDMAX,KUTTA
      COMMON /STEP/ NITTOT,TIMETOT,DISTTOT,NITELM,TIMEELM,DISTELM,DISTO
     &              ,XGSTEP(3),DXSTEP,DSSTEP,DISCRQ,NCHK,BNDMAX,KUTTA
     &              ,DISTMNQ,XGODIST(3)
C----------END STEP_COM---------
C-----COMMON START_COM--------
      INTEGER DIMSTA
      LOGICAL RDNODE,KWNODE
      COMMON /START/ NTRAJ,MAXSTEP,NRINTR,IELSTA,XGSTART(3),XLSTART(3)
     &               ,DIMSTA,MXTRAJ
     &               ,RDNODE,KWNODE,MXNODE,NBNODE
C----------END START_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &               , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
      COMMON /ORIGIN/ XORIG(3)


C-----COMMON ARRIVE_COM------
      PARAMETER (DEFRARR=1D0)
      LOGICAL ATDSTR,ARRIVED
      CHARACTER FIATD*256
      COMMON /ARRIVE/ ATDSTR,XARRIVE(3),RARRIVE
     &     ,TARRIVE,DARRIVE,IARRIVE,FIATD,ARRIVED
C----------END ARRIVE_COM-------

      LOGICAL INS,HALT,WROUT,MOVED
      CHARACTER*(*) MESSAGE
      DIMENSION XL(3),XG(3),QL(3),QG(3),XLNEW(3),XGNEW(3),DXG(3),DXL(3)

C---------------   INITIAL VALUES -------------------------------------

C -- Value of DSSTEP is changed (finer) if RUNGE is not used
      INS=.TRUE.
      HALT=.FALSE.
      ID=ICYCLE(IPBND,IELPOS,MXBOUND)
      IEL=IELBND(ID)
      KR=KRBND(ID)
      DSSTEPL=STSIZE(KR)
      IF (BNDFLOW(ID)) DSSTEPL=DSSTEPL/SUBSTEP
      N=NDIM(KR)
      DT=0D0
      DS=0D0
      CALL VEQV(3,XG,XGBND)
      CALL VEQV(N,XL,XLBND(1,ID))
      CALL VEQV(N,QL,QLBND(1,ID))
      CALL VEQV(3,QG,QGBND(1,ID))
      TIMELOG=0D0
      IF (TIMETOT.GT.0D0) TIMELOG=LOG10(TIMETOT)
      ICOLOR=NELP(IEL)
      ICOLOR=ICOLOR-MOD(ICOLOR,10)
      IF(ICOLOR.GT.200) ICOLOR=ICOLOR-MOD(ICOLOR,200)

C--------------   BEGIN OF STEP-ITERATION  ----------------------------

      DO 1000 NIT=2,MAXITER+1
```

```
C Check distance if output should be written
        CALL CHKDIST(NITTOT,DISTMNQ,XGODIST,XG,WROUT,MOVED)
        IF (MOVED) NITTOT=NITTOT+1
        WROUT=(WROUT.OR..NOT.INS)
        IF (WROUT.AND.MOVED) THEN
          WRITE (OUPAT,'(I4,3(3X,F12.3),I5)')
     &         NITTOT,(XG(I)+XORIG(I),I=1,3),ICOLOR
          WRITE (OULVT,'(I4,1P,3(3X,E12.5),I5)')
     &         NITTOT,DISTTOT,TIMETOT,TIMELOG,ICOLOR
          CALL VEQV(3,XGODIST,XG)
        ENDIF

        IF (NITTOT.GT.MAXSTEP) GOTO 1600
        CALL CHKSTOP(XG,*1500,*1700)

C --Check if particle arrived at specified location
        IF (ATDSTR)
     &    CALL CHKARR(XG,DS,DT)

        IF (HALT) THEN
          CALL VEQV(3,XGBND,XG)
          CALL VEQV(3,QGBND(1,ID),QG)
          CALL VEQV(N,XLBND(1,ID),XL)
          CALL VEQV(N,QLBND(1,ID),QL)
          IFAIL=0
          OUTOFEL=.NOT.INS
          RETURN
        ENDIF
        IF (NIT.GT.MAXITER) GOTO 1100
C----------------  CALCULATING NEW POINT ------------

        QLABS=VABS(N,QL)
        IF (.NOT.(QLABS.GT.0.)) GOTO 1200
        DT=DSSTEPL/QLABS
        IF (.NOT.BNDFLOW(ID).AND.KUTTA) THEN
          CALL RUNGE(IEL,XL,DT,QL,QG,XKRBND(1,1,ID),PERMBND(1,1,ID)
     &      ,PORBND(ID),HEADBND(1,ID),KR,N,IERR)
          IF (IERR.NE.0) GOTO 1300
        ENDIF

        DO 100 I=1,N
  100   XLNEW(I)=XL(I)+QL(I)*DT

C --- I/O TEST ; GLOBAL VALUES ; NEW VELOCITIES ---

        CALL IOTEST(XL,XLNEW,XLNEW,KR,NAEXBND(1,ID),NBOUND(ID)
     &     ,INS,.TRUE.,.TRUE.,.TRUE.,CHKBND,BNDFLOW(ID)
     &     ,FLOWDIM,NAEXP(ID),IERR)

        HALT=((.NOT.INS).OR.ONESTEP)

        CALL VELOTXT(IEL,XLNEW,XKRBND(1,1,ID),PERMBND(1,1,ID)
     &   ,PORBND(ID),HEADBND(1,ID),KR,N,QL,QG,FMBND(1,ID)
     &   ,FMIBND(1,ID),IERR,'STEPELM')
        IF(IERR.NE.0) GOTO 1400

        CALL SHAPEV(XLNEW,XGNEW,XKRBND(1,1,ID),KR)
        CALL VDIF(3,XGNEW,XG,DXG)
        DS=VABS(3,DXG)
        IF (.NOT.INS) THEN
          CALL VDIF(N,XLNEW,XL,DXL)
          DT=(VABS(N,DXL)/QLABS)
        ENDIF
        CALL VEQV(N,XL,XLNEW)
        CALL VEQV(3,XG,XGNEW)

        NITELM=NIT
        TIMEELM=TIMEELM+DT/YEAR
        DISTELM=DISTELM+DS
        DISTTOT=DISTTOT+DS
        TIMETOT=TIMETOT+DT/YEAR
        TIMELOG=0D0
        IF(TIMETOT.GT.0D0) TIMELOG=LOG10(TIMETOT)
```

```
 1000 CONTINUE

C Error messages
1100  IFAIL=10
      WRITE(MESSAGE,'(A)')
     &  'ERROR (STEPELM): Too many steps for this element'
      RETURN
1200  IFAIL=20
      WRITE(MESSAGE,'(A)')
     &  'Track stopped (FLUX IS ZERO)'
      RETURN
1300  IFAIL=30
      WRITE(MESSAGE,'(A)')
     &  'ERROR (STEPELM): Failure from subroutine RUNGE'
      RETURN
1400  IFAIL=40
      WRITE(MESSAGE,'(A)')
     &  'ERROR (STEPELM): Failure from subroutine VELOX'
      RETURN
1500  IFAIL=50
      WRITE(MESSAGE,'(A)')
     &  'Track concentrates within small region'
      RETURN
1600  IFAIL=60
      WRITE(MESSAGE,'(A)') 'Limit of maximum steps reached'
      RETURN
1700  IFAIL=70
      WRITE(MESSAGE,'(A)')
     &  'Track oscillating'
      RETURN
C----------END STEPELM----------
      END



      FUNCTION STSIZE(KR)
      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON STEP_COM-----------

      PARAMETER (MAXITER=100,DFSTEP=1D-1,SUBRA=5D-4
     &          ,SUBSTEP=2D0,NCHECK=10,DFDSCR=1D-1
     &          ,DEFDIST=1D-2)

      LOGICAL   BNDMAX,KUTTA
      COMMON /STEP/ NITTOT,TIMETOT,DISTTOT,NITELM,TIMEELM,DISTELM,DISTO
     &             ,XGSTEP(3),DXSTEP,DSSTEP,DISCRQ,NCHK,BNDMAX,KUTTA
     &             ,DISTMNQ,XGODIST(3)
C----------END STEP_COM---------
      STSIZE=DSSTEP
      IF(KR.EQ.6 .OR. KR.EQ.10 .OR. KR.EQ.15) STSIZE=5D-1*DSSTEP
      RETURN
C----------END STSIZE-----------
      END



      SUBROUTINE CHKSTOP(XG,*,*)

      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON STEP_COM-----------

      PARAMETER (MAXITER=100,DFSTEP=1D-1,SUBRA=5D-4
     &          ,SUBSTEP=2D0,NCHECK=10,DFDSCR=1D-1
     &          ,DEFDIST=1D-2)

      LOGICAL   BNDMAX,KUTTA
      COMMON /STEP/ NITTOT,TIMETOT,DISTTOT,NITELM,TIMEELM,DISTELM,DISTO
     &             ,XGSTEP(3),DXSTEP,DSSTEP,DISCRQ,NCHK,BNDMAX,KUTTA
     &             ,DISTMNQ,XGODIST(3)
C----------END STEP_COM---------
      DIMENSION DX(3),XG(3)

      IF(NITTOT.LE.1) THEN
         NCHK=0
         CALL VEQV(3,XGSTEP,XG)
```

```
         DISTO=DISTTOT
         RETURN
      ENDIF

      NCHK=NCHK+1
      IF(NCHK.LT.NCHECK) RETURN

      CALL VDIF(3,XGSTEP,XG,DX)
      DELTASQ=(0.3*(DISTTOT-DISTO))**2
      DXQ=VSCP(3,DX,DX)
      IF (DXQ.LT.DELTASQ) RETURN 2     !* Oscillation
      IF (DXQ.LT.DISCRQ) RETURN 1      !* Concentration in small region
      IF(NCHK.GT.NCHECK) THEN
         CALL VEQV(3,XGSTEP,XG)
         DISTO=DISTTOT
         NCHK=0
      ENDIF

      RETURN
C----------END CHKSTOP----------
      END


      SUBROUTINE CHKDIST(NITTOT,DISTQ,XOLD,XNEW,GREATER,MOVED)

      IMPLICIT REAL*8 (A-H,O-Z)

      PARAMETER (DXQMOVE=1D-7)
      LOGICAL GREATER,MOVED

      DIMENSION XOLD(3),XNEW(3),DX(3)

      IF (NITTOT.GT.1) THEN
        CALL VDIF(3,XNEW,XOLD,DX)
        DXQ=VSCP(3,DX,DX)
        GREATER=(DXQ.GT.DISTQ)
        MOVED=(DXQ.GT.DXQMOVE)
      ELSE
        MOVED=.TRUE.
        GREATER=.TRUE.
      ENDIF

      RETURN
C----------END CHKDIST----------
      END


      SUBROUTINE CHKARR(XG,DELM,TELM)

      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON ARRIVE_COM------
      PARAMETER (DEFRARR=1D0)
      LOGICAL ATDSTR,ARRIVED
      CHARACTER FIATD*256
      COMMON /ARRIVE/ ATDSTR,XARRIVE(3),RARRIVE
     &     ,TARRIVE,DARRIVE,IARRIVE,FIATD,ARRIVED
C----------END ARRIVE_COM-------
C-----COMMON STEP_COM-----------

      PARAMETER (MAXITER=100,DFSTEP=1D-1,SUBRA=5D-4
     &          ,SUBSTEP=2D0,NCHECK=10,DFDSCR=1D-1
     &          ,DEFDIST=1D-2)

      LOGICAL   BNDMAX,KUTTA
      COMMON /STEP/ NITTOT,TIMETOT,DISTTOT,NITELM,TIMEELM,DISTELM,DISTO
     &              ,XGSTEP(3),DXSTEP,DSSTEP,DISCRQ,NCHK,BNDMAX,KUTTA
     &              ,DISTMNQ,XGODIST(3)
C----------END STEP_COM---------

      DIMENSION XG(3),DX(3)

      IF (ARRIVED) RETURN
```

```
      CALL VDIF(3,XG,XARRIVE,DX)
      ARRIVED=(VSCP(3,DX,DX).LT.RARRIVE)
      IF (.NOT.ARRIVED) RETURN
      IARRIVE=1
      TARRIVE=TIMETOT+TELM
      DARRIVE=DISTTOT+DELM
      RETURN
C----------END CHKARR----------
      END


      SUBROUTINE WRELEM(IELM,NELP)

      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &     FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &    ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &    ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &     BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &    ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &    ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &     XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &    ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &    ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &    ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &    ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------
C-----COMMON STEP_COM-----------

      PARAMETER (MAXITER=100,DFSTEP=1D-1,SUBRA=5D-4
     &          ,SUBSTEP=2D0,NCHECK=10,DFDSCR=1D-1
     &          ,DEFDIST=1D-2)

      LOGICAL   BNDMAX,KUTTA
      COMMON /STEP/ NITTOT,TIMETOT,DISTTOT,NITELM,TIMEELM,DISTELM,DISTO
     &             ,XGSTEP(3),DXSTEP,DSSTEP,DISCRQ,NCHK,BNDMAX,KUTTA
     &             ,DISTMNQ,XGODIST(3)
C----------END STEP_COM---------

      CHARACTER*50 FMT(3)

      DATA FMT(3)(1:15) /'(3I7,1P,6E15.5)'/
     &    ,FMT(2)(1:14) /'(3I7,1P,6E15.5'/
     &    ,FMT(2)(15:37) /',5X,''(** 2D TRACK **)'')'/
     &    ,FMT(1)(1:14) /'(3I7,1P,6E15.5'/
     &    ,FMT(1)(15:37) /',5X,''(** 1D TRACK **)'')'/

      VELAVR=0D0
      VELMEAN=0D0
      IF (TIMEELM.GT.0D0) VELAVR=DISTELM/TIMEELM
      IF (TIMETOT.GT.0D0) VELMEAN=DISTTOT/TIMETOT

      WRITE (OUTRA,FMT(FLOWDIM)) IELM,NELP,NITTOT,DISTELM
     &       ,TIMEELM,VELAVR,DISTTOT,TIMETOT,VELMEAN
```

```
      TIMEELM=0D0
      DISTELM=0D0

      RETURN
C----------END WRELEM-----------
      END


      SUBROUTINE IOTEST(X1,X2,XE,KR,NAEX,NBOUND,INS,EXA,EXC,BNDXE
     &                  ,BNDX2,BNDFLOW,FLOWDIM,NAEXP,IERR)
C-----------------IOTEST-----------------------
C              --VARIABLES--
C
C        X2 :    NEW LOCAL COORDINATES
C        X1 :    OLD L. C.
C    L1,L2  :    AEREA (VOLUME-)COORDINATES
C        XE :    L. C. OF LEAVING POINT
C        DX :    DIFFERENCE X2-X1
C        DL :         "      L2-L1
C       FAC :    FACTOR SUCH THAT X1+FAC*DX=SIGNUM(DX)
C        KR :    NUMBER OF NODES (ELEMENTTYPE)
C      NAEX :    NUMBER OF AREA (SEE FOR CONVENTION)
C       INS :    LOGICAL PARAMETER
C             FALSE    : POINT IS OUTSIDE
C             TRUE     : POINT IS INSIDE
C       EXA:     LOGICAL
C       ----  TRUE     :CALCULATING LEAVING AREA
C            FALSE     :ONLY I/O-TEST
C
C       EXC:
C       ----  TRUE     :CALCULATING LEAVING COORDINATES
C            FALSE     :OPPOSITE
C       BNDXE:
C       ----  TRUE     :CHECKING BOUNDARIES OF XE
C            FALSE     :OPPOSITE
C       BNDX2:
C       ----  TRUE     :CHECKING BOUNDARIES OF X2
C            FALSE     :OPPOSITE
C       BNDFLOW  :
C       ------ TRUE    :STEP ALONG BOUNDARY (INSIDE FOR THIS BOUNDARY)
C
C       FLOWDIM  :      DIMENSION OF BOUNDARY ALONG WHICH THE STEP WAS MADE
C       -------
C       NAEXP    :      UNIQUE NUMBER THAT REPRESENTS THE FLOW BOUNDARIES
C       -----
C       IERR:
C       ----    0       :OKAY
C              10       :INVALID ELEMENTTYPE (i.e.KR UNKNOWN)
C              20       :X1 IS OUTSIDE THE ELEMENT (LEAVING COORDINATES
C                   ARE CALCULATED WITH LSTART(=MCP)
C-------------------------------------------------------------------------
C-------------------------------------------------------------------------
C      TEST OF WETHER A POINT IS INSIDE OR OUTSIDE OF AN ELEMENT
C      CALCULATION OF THE EXIT AREA
C      CALCULATION OF THE LEAVING POINT
C      CHECKING IF THE LEAVING POINT LIES ON MORE BOUNDARIES
C-------------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L1(6),L2(6),LE(6)
      INTEGER FLOWDIM

      DIMENSION X1(*),X2(*),XE(*),NAEX(*),XL0(3),NAEXDUM(2)
      LOGICAL INS,EXA,EXC,BNDX2,BNDXE,INS2(6),INS1(6),KRTEST,ANDV
     &       ,GOON,BNDFLOW

      IF (.NOT.KRTEST(KR)) THEN
        IERR=10
        RETURN
      ENDIF
      IERR=0
      NA=NUMAREA(KR)

C   --- INSIDE/OUTSIDE ---
```

```
      CALL VOLCOR(KR,X2,L2,.TRUE.)
      DO 100 I=1,NA
  100 INS2(I)=(L2(I).GE.0D0)

C Point is inside if step was made along a boundary
      IF (BNDFLOW) THEN
        NB=NDIM(KR)-FLOWDIM
        IF (NB.EQ.1) THEN
          NAEXDUM(1)=NAEXP
        ELSE
          NAEXDUM(1)=NAEXP/10
          NAEXDUM(2)=MOD(NAEXP,10)
        ENDIF
        DO 150 I=1,NB
  150   INS2(NAEXDUM(I))=.TRUE.
      ENDIF

      INS=ANDV(NA,INS2)
      GOON=(((.NOT.INS).AND.(EXA.OR.EXC)).OR.BNDX2)
      IF (.NOT.GOON) RETURN

C   --- EXITING AREA ---

      CALL VOLCOR(KR,X1,L1,.TRUE.)
      DO 200 I=1,NA
  200 INS1(I)=(L1(I).GE.0D0)
      IF (.NOT.ANDV(NA,INS1)) THEN
        IERR=20
        CALL LSTART(KR,XL0)
        CALL VOLCOR(KR,XL0,L1,.TRUE.)
      ENDIF

      IF (INS) THEN
        CALL VEQV(6,LE,L2)
        NAEX(1)=0
      ELSE
        CALL EXAREA(NA,INS2,L1,L2,LE,NAEX(1),EXC)
        IF (EXC) CALL VOLCOR(KR,XE,LE,.FALSE.)
      ENDIF
      GOON=(BNDXE.OR.BNDX2)
      IF (.NOT.GOON) RETURN
C   --- CHECK BOUNDS OF LEAVING POINT ---

      CALL BNDTEST(KR,NA,LE,NAEX,NBOUND)

      RETURN
C----------END IOTEST-----------
      END


      SUBROUTINE VOLCOR(KR,X,L,XTOL)

      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L(6),X(*)

      LOGICAL XTOL

      IF (KR.EQ.3)                GOTO 100
      IF (KR.EQ.6)                GOTO 200
      IF (KR.EQ.8.OR.KR.EQ.9)     GOTO 300
      IF (KR.EQ.10)               GOTO 400
      IF (KR.EQ.13.OR.KR.EQ.14)   GOTO 500
      IF (KR.EQ.15.OR.KR.EQ.18)   GOTO 600
      IF (KR.EQ.20.OR.KR.EQ.27)   GOTO 700
      RETURN

  100 CONTINUE
C     -------- KR = 3 --------------------
      IF (XTOL) THEN
        L(1)=1D0+X(1)
        L(2)=1D0-X(1)
        DO 110 I=3,6
  110   L(I)=0D0
```

```
      ELSE
        X(1)=L(1)-1D0
      ENDIF
      RETURN

  200 CONTINUE
C     -------- KR = 6 --------------------
      IF (XTOL) THEN
        L(2)=X(1)
        L(3)=X(2)
        L(1)=1D0-L(2)-L(3)
        DO 210 I=4,6
  210   L(I)=0D0
      ELSE
        X(1)=L(2)
        X(2)=L(3)
      ENDIF
      RETURN

  300 CONTINUE
C     -------- KR = 8/9 ------------------
      IF (XTOL) THEN
        L(1)=1D0+X(2)
        L(2)=1D0-X(1)
        L(3)=1D0-X(2)
        L(4)=1D0+X(1)
        DO 310 I=5,6
  310   L(I)=0D0
      ELSE
        X(1)=1D0-L(2)
        X(2)=1D0-L(3)
      ENDIF
      RETURN

  400 CONTINUE
C     -------- KR = 10 -------------------
      IF (XTOL) THEN
        L(1)=X(3)
        L(2)=1D0-X(1)-X(2)-X(3)
        L(3)=X(1)
        L(4)=X(2)
      ELSE
        X(1)=L(3)
        X(2)=L(4)
        X(3)=L(1)
      ENDIF
      RETURN

  500 CONTINUE

C     -------- KR = 13/14 -----------------
      IF (XTOL) THEN
        L(1)=X(3)+1D0
        L(2)=1D0+X(2)
        L(3)=1D0-X(1)
        L(4)=1D0-X(2)
        L(5)=1D0+X(1)
      ELSE
        X(3)=L(1)-1D0
        X(1)=1D0-L(3)
        X(2)=L(2)-1D0
      ENDIF
      RETURN

C     -------- KR = 13/14 -----------------
C       IF (XTOL) THEN
C         L(1)=X(3)+1D0
C         L(2)=X(2)+1D0
C         L(3)=-X(1)-X(3)
C         L(4)=-X(2)-X(3)
C         L(5)=1D0+X(1)
C       ELSE
C         X(3)=L(1)-1D0
C         X(1)=-L(3)-X(3)
C         X(2)=L(2)-1D0
```

```
C       ENDIF
C       RETURN

  600 CONTINUE
C     -------- KR = 15/18 -----------------
      IF (XTOL) THEN
        L(1)=1D0+X(3)
        L(2)=1D0-X(3)
        L(3)=1D0-X(1)-X(2)
        L(4)=X(1)
        L(5)=X(2)
      ELSE
        X(1)=L(4)
        X(2)=L(5)
        X(3)=1D0-L(2)
      ENDIF
      RETURN

  700 CONTINUE
C     -------- KR = 20/27-----------------
      IF (XTOL) THEN
        L(1)=1D0-X(1)
        L(2)=1D0+X(1)
        L(3)=1D0-X(2)
        L(4)=1D0+X(2)
        L(5)=1D0-X(3)
        L(6)=1D0+X(3)
      ELSE
        X(1)=1D0-L(1)
        X(2)=1D0-L(3)
        X(3)=1D0-L(5)
      ENDIF
      RETURN

C----------END VOLCOR-----------
      END


      SUBROUTINE EXAREA(NA,INS2,L1,L2,LE,NAEX,EXC)

      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L1(*),L2(*),LE(*),DL(6),FACT(6)

      LOGICAL INS2(*),EXC

      FACTMN=1.D20
      NAEX=0
      DO 100 I=1,NA
        DL(I)=L2(I)-L1(I)
        IF (.NOT.INS2(I).AND.((DL(I).GT.0D0).OR.(DL(I).LT.0D0))) THEN
          FACT(I)=-L1(I)/DL(I)
          IF (FACT(I).LT.FACTMN) THEN
            FACTMN=FACT(I)
            NAEX=I
          ENDIF
        ENDIF
  100 CONTINUE

      IF (NAEX.EQ.0)
     &  CALL ABORT('CALCULATION OF LEAVING COORDINATES FAILED DURING '//
     &    'CALL OF SUBROUTINE EXAREA (TEL. 01/2565105)')

      IF (.NOT.EXC) RETURN
      DO 200 I=1,NA
        LE(I)=L1(I)+FACTMN*DL(I)
        IF (LE(I).LT.0D0) LE(I)=0D0
  200 CONTINUE
      LE(NAEX)=0D0

      RETURN
C----------END EXAREA-----------
      END
```

```
      SUBROUTINE BNDTEST(KR,NA,LE,NAEX,NBOUND)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (EPS=1D-5)
      REAL*8 LE(*)
      DIMENSION NAEX(*)

      IF (NAEX(1).EQ.0) THEN
        NBOUND=0
      ELSE
        NBOUND=1
      ENDIF

      DO 100 I=1,NA
        IF (I.EQ.NAEX(1)) GOTO 100
        IF (LE(I).LT.EPS) THEN
          NBOUND=NBOUND+1
          NAEX(NBOUND)=I
        ENDIF
  100 CONTINUE
      CALL ISORT(NBOUND,NAEX,NAEX)

C -- PYRAMID HANDLING --

      IF ((NBOUND.EQ.3).AND.(KR.EQ.13.OR.KR.EQ.14)
     &                          .AND.(NAEX(1).NE.1)) THEN
        NBOUND=4
        DO 200 I=1,4
  200   NAEX(I)=I+1
      ENDIF

      DO 300 I=NBOUND+1,4
  300 NAEX(I)=100

      RETURN
C----------END BNDTEST----------
      END


      SUBROUTINE NEXTELM(*,MESSAGE)
C---------------------------------------
C     GET NEXT ELEMENT (RETURN 1 IF NONE)
C---------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &   ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
```

```
     &    ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &     XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &    ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &    ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &    ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &    ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------
      CHARACTER*(*) MESSAGE

C NEW NEIGHBOURS IF BOUNDARY WAS CHECKED (IN SUBR. IOTEST)

      ID=ICYCLE(IPBND,IELPOS,MXBOUND)
      BNDDIM=MAX(NDIM(KRBND(ID))-NBOUND(ID),0)
      IF (OUTOFEL.OR.CHKBND) THEN

C check boundaries for pinch sided element
        IF (NARV(IELBND(ID)).LT.0)
     &    CALL CHKPNCH(ID)

        IF (BNDDIM.EQ.MAXDIM-1) THEN
          CALL CONELM    !* 1 boundary --> ICON
        ELSE
          CALL ALLELM(*1000) !* (>1) boundary --> all elements
        ENDIF
        CALL CHKOUTS(*1100)
      ENDIF

      CALL NXTCOOR(*1200,IEL,ID)
      CALL NXTVELO(IEL,ID)
      CALL VELOTST(*1400)
      CALL MAXVELO
      RETURN

C  ----- NEXT ELEMENT NOT FOUND -----

 1000 MESSAGE='-- ERROR --. Too many elements for COMMON BOUND'
      RETURN 1
 1100 MESSAGE='Track left region normally'
      RETURN 1
1200  WRITE(MESSAGE,'(A,I5,A,I2,A)')
     &    'ERROR (NXTCOOR): local entry point not found for element ',
     &     IELM(IEL),' (KR=',KRV(IEL),')'
      RETURN 1
 1300 MESSAGE='ERROR (NXTVELO): next entry point not found'
      IELFAIL=IEL
      IDFAIL=ID
      RETURN 1
 1400 MESSAGE='Track ends in corner node'
      RETURN 1

C----------END NEXTELM----------
      END



      SUBROUTINE CHKPNCH(ID)
C ---------------------------------------------------------------------
C Check boundaries for pinch sided elements,since the local coordinates
C may differ from boundary values.
C Therefore you have to check in all cases the corners.If a boundary is found
C the right combination of NAEX(1).. NAEX(NBOUND) must be placed.
C ---------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
```

```
     &           ,MXLM,MNNIC,MXNIC,MAXDIM
     &           ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &   ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &   ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &    XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &   ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &   ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &   ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &   ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------

      PARAMETER (EPSILON=1D-4,EPSLQ=EPSILON**2)

      INTEGER CORNER,EDGENOD

      DIMENSION XL(3),XG(3),NODES(9),CORNER(0:4),MIDSIDE(4)
     &         ,DXG(3),XGEDGE(3,3),EDGENOD(3,4),XL0(3)

      DATA CORNER(0) /0/  !* is made to check easily equality of corners

      KR=KRBND(ID)
      N=NDIM(KR)
      CALL VEQV(N,XL,XLBND(1,ID))
      CALL VEQV(3,XG,XGBND)

C get one boundary face and corresponding nodes
      NAEX=NAEXBND(1,ID)
      CALL FACENOD(KR,NAEX,NODES,NPERFAC)
      NODES(NPERFAC+1)=NODES(1) !* is made to get easily the edges

C get corners and midsides and edges
      NEDG=(NPERFAC+1)/2
      DO 100 I=1,NEDG
        IC=2*I-1
        IM=2*I
        CORNER(I)=NODES(IC)
        MIDSIDE(I)=NODES(IM)
        DO 99 J=1,3
   99   EDGENOD(J,I)=NODES(IC+J-1)
  100 CONTINUE

C get global error out of EPSILON
      DXGQ=-1D0
      DO 200 I=2,KR
        CALL VDIF(3,XKRBND(1,1,ID),XKRBND(1,I,ID),DXG)
        DXGQ=MAX(DXGQ,VSCP(3,DXG,DXG))
  200 CONTINUE
      EPSG=SQRT(DXGQ)*EPSILON
      EPSGQ=DXGQ*EPSLQ
```

```
C Check if global point lies in a corner
C Loop over each corner and check for XG
      DO 300 I=1,NEDG

C   skip equal corners (already treated)
        IF (CORNER(I).EQ.CORNER(I-1)) GOTO 300

        CALL VDIF(3,XKRBND(1,CORNER(I),ID),XG,DXG)
        IF (VSCP(3,DXG,DXG).GT.EPSGQ) GOTO 300
        CALL GETNAEX(KR,CORNER(I),NBOUND(ID),NAEXBND(1,ID))
        BNDDIM=0
        RETURN
  300 CONTINUE


C No corner found;check edges
C   get edge coordinates for each edge and check for XG
      DO 400 I=1,NEDG

C       skip pinched edges (because they are corners)
        IF (EDGENOD(1,I).EQ.EDGENOD(3,I)) GOTO 400

        DO 350 J=1,3
  350   CALL VEQV(N,XGEDGE(1,J),XKRBND(1,EDGENOD(J,I),ID))

        CALL LSTART(3,XL0)
        CALL SEARCH(XG,XL,XL0,3,XGEDGE,MAXDIM,EPSG,IFAIL)
        IF (IFAIL.NE.0) GOTO 400
        CALL GETNAEX(KR,MIDSIDE(I),NBOUND(ID),NAEXBND(1,ID))
        BNDDIM=1
        RETURN
  400 CONTINUE

C No edge found;boundaries are not changed
      IF (NBOUND(ID).GT.1) THEN
        WRITE (OUTRA,11000) KR,IELM(IELBND(ID)),NBOUND(ID),BNDDIM
     &             ,(NAEXBND(J,ID),J=1,4),(XLBND(J,ID),J=1,3)
      ENDIF

      RETURN

11000 FORMAT (' WARNING FROM "CHKPNCH".BOUNDARIES ARE NOT CORRECT'
     &       /' Element Type:  KR =',I8
     &       /' Element No:  IELM =',I8
     &       /' No. of Boundaries:  NBOUND =',I8
     &       /' Boundary Dimension:  BNDDIM =',I8
     &       /' Boundary Faces:  NAEXBND =',4I5
     &       /' Local Coordinates: XL(1..3) =',3F12.8 )

C----------END CHKPNCH----------
      END


      SUBROUTINE FACENOD(KR,NAEX,NODES,NEDGE)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NODES(8),ITYPE(20),NNODES(8,6,7),NPERFAC(6,7)

      DATA ITYPE /0,0,1,0,0,2,0,3,0,4,0,0,5,0,6,0,0,0,0,7/
      DATA ((NNODES (I,J,1),I=1,8),J=1,6)
     &             / 1, 0, 0, 0, 0, 0, 0, 0,   !* line face 1
     &               3, 0, 0, 0, 0, 0, 0, 0,   !* line face 2
     &               1, 2, 3, 0, 0, 0, 0, 0,   !* line face 3
     &               0, 0, 0, 0, 0, 0, 0, 0,   !* dummy face 4
     &               0, 0, 0, 0, 0, 0, 0, 0,   !* dummy face 5
     &               0, 0, 0, 0, 0, 0, 0, 0/   !* dummy face 6

      DATA ((NNODES (I,J,2),I=1,8),J=1,6)
     &             / 1, 2, 3, 0, 0, 0, 0, 0,   !* triangle face 1
     &               3, 4, 5, 0, 0, 0, 0, 0,   !* triangle face 2
     &               5, 6, 1, 0, 0, 0, 0, 0,   !* triangle face 3
     &               1, 2, 3, 4, 5, 6, 0, 0,   !* triangle face 4
     &               1, 2, 3, 4, 5, 6, 0, 0,   !* triangle face 5
```

```
     &              0, 0, 0, 0, 0, 0, 0, 0/  !* dummy face 6

      DATA ((NNODES (I,J,3),I=1,8),J=1,6)
     &             / 1, 2, 3, 0, 0, 0, 0, 0,  !* rectangle face 1
     &               3, 4, 5, 0, 0, 0, 0, 0,  !* rectangle face 2
     &               5, 6, 7, 0, 0, 0, 0, 0,  !* rectangle face 3
     &               7, 8, 1, 0, 0, 0, 0, 0,  !* rectangle face 4
     &               1, 2, 3, 4, 5, 6, 7, 8,  !* rectangle face 5
     &               1, 2, 3, 4, 5, 6, 7, 8/  !* rectangle face 6

      DATA ((NNODES (I,J,4),I=1,8),J=1,6)
     &             / 1, 2, 3, 4, 5, 6, 0, 0,  !* tetra face 1
     &               1, 2, 3, 8,10, 7, 0, 0,  !* tetra face 2
     &               3, 4, 5, 9,10, 8, 0, 0,  !* tetra face 3
     &               5, 6, 1, 7,10, 9, 0, 0,  !* tetra face 4
     &               0, 0, 0, 0, 0, 0, 0, 0,  !* dummy face 5
     &               0, 0, 0, 0, 0, 0, 0, 0/  !* dummy face 6

      DATA ((NNODES (I,J,5),I=1,8),J=1,6)
     &             / 1, 2, 3, 4, 5, 6, 7, 8,  !* pyram face 1
     &               1, 2, 3,10,13, 9, 0, 0,  !* pyram face 2
     &               3, 4, 5,11,13,10, 0, 0,  !* pyram face 3
     &               5, 6, 7,12,13,11, 0, 0,  !* pyram face 4
     &               7, 8, 1, 9,13,12, 0, 0,  !* pyram face 5
     &               0, 0, 0, 0, 0, 0, 0, 0/  !* dummy face 6

      DATA ((NNODES (I,J,6),I=1,8),J=1,6)
     &             / 1, 2, 3, 4, 5, 6, 0, 0,  !* prism face 1
     &              10,11,12,13,14,15, 0, 0,  !* prism face 2
     &               1, 2, 3, 8,12,11,10, 7,  !* prism face 3
     &               3, 4, 5, 9,14,13,12, 8,  !* prism face 4
     &               5, 9,14,15,10, 7, 1, 6,  !* prism face 5
     &               0, 0, 0, 0, 0, 0, 0, 0/  !* dummy face 6

      DATA ((NNODES (I,J,7),I=1,8),J=1,6)
     &             / 3, 4, 5,11,17,16,15,10,  !* brick face 1
     &               1, 8, 7,12,19,20,13, 9,  !* brick face 2
     &               5, 6, 7,12,19,18,17,11,  !* brick face 3
     &               1, 2, 3,10,15,14,13, 9,  !* brick face 4
     &              13,14,15,16,17,18,19,20,  !* brick face 5
     &               1, 2, 3, 4, 5, 6, 7, 8/  !* brick face 6

      DATA (NPERFAC(I,1),I=1,6) /1,1,3,0,0,0/
      DATA (NPERFAC(I,2),I=1,6) /3,3,3,6,6,0/
      DATA (NPERFAC(I,3),I=1,6) /3,3,3,3,8,8/
      DATA (NPERFAC(I,4),I=1,6) /6,6,6,6,0,0/
      DATA (NPERFAC(I,5),I=1,6) /8,6,6,6,6,0/
      DATA (NPERFAC(I,6),I=1,6) /6,6,8,8,8,0/
      DATA (NPERFAC(I,7),I=1,6) /8,8,8,8,8,8/

      IF (NAEX.GE.1.AND.NAEX.LE.8.AND.ITYPE(KR).GT.0) THEN
         IELTYPE=ITYPE(KR)
         CALL IVEQV(8,NODES,NNODES(1,NAEX,IELTYPE))
         NEDGE=NPERFAC(NAEX,IELTYPE)
      ENDIF
      RETURN
C----------END FACENOD----------
      END


      SUBROUTINE CONELM
C-------------------------------------------------------
C     GET ELEMENTS CONNECTED THROUGH ICON
C-------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
```

```
     &            ,MXLM,MNNIC,MXNIC,MAXDIM
     &            ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &        ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &     FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &    ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &    ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &     BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &    ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &    ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &     XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &    ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &    ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &    ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &    ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------

      LOGICAL BORD
      DIMENSION ID(3),IEL(3),KR(3),ND(3),NAEX(3)

C   Initial values
      ID(1)=ICYCLE(IPBND,IELPOS,MXBOUND)
      IEL(1)=IELBND(ID(1))
      KR(1)=KRBND(ID(1))
      ND(1)=NDIM(KR(1))
      IPBND=ICYCLE(IPBND,IELPOS-1,MXBOUND)
      IELPOS=1
      IF (NAEXBND(1,ID(1)).EQ.100)
     &    NAEXBND(1,ID(1))=NUMAREA(KR(1))+1
      NAEX(1)=NAEXBND(1,ID(1))
      IEL(2)=ICON(NAEX(1),IEL(1))
      BORD=(IEL(2).EQ.-1)
      ID(2)=ICYCLE(ID(1),1,MXBOUND)
      IF (.NOT.BORD) THEN
         KR(2)=KRV(IEL(2))
         ND(2)=NDIM(KR(2))
      ENDIF

C Distinguish 4 cases
C  1.:  1. Element (max. dim.) on border
C  2.:  2 Elements of max. dim
C  3.:  3 Elements (1. of max. dim.)
C  4.:  3 Elements (1. of low dim.)

      IF (ND(1).EQ.MAXDIM.AND.BORD) THEN
         NNGHB=1
         RETURN
      ELSE
         IF (.NOT.BORD)
     &     CALL NEIGHB(IEL(1),IEL(2),ICON(1,IEL(2)),NAEX(2))
         IF (ND(1).EQ.MAXDIM.AND.ND(2).EQ.MAXDIM) THEN
            NNB=2
         ELSEIF (ND(1).EQ.MAXDIM.AND.ND(2).EQ.2) THEN
            NA=NUMAREA(KR(2))
            N1=NAEX(2)
            N2=ICYCLE(N1-NA,1,2)+NA
            IEL(3)=ICON(N2,IEL(2))
            IF (IEL(3).EQ.-1) THEN
               NNB=2
            ELSE
```

```
               CALL NEIGHB(IEL(2),IEL(3),ICON(1,IEL(3)),NAEX(3))
               NNB=3
            ENDIF
         ELSEIF (ND(1).EQ.MAXDIM-1) THEN
            NA=NUMAREA(KR(1))
            N1=NAEX(1)
            N2=ICYCLE(N1-NA,1,2)+NA
            IEL(3)=ICON(N2,IEL(1))
            NNB=1
            DO 100 I=2,3
               IF (IEL(I).NE.-1) THEN
                  NNB=NNB+1
                  CALL NEIGHB(IEL(1),IEL(I),ICON(1,IEL(I)),NAEX(I))
               ENDIF
  100       CONTINUE
         ENDIF
      ENDIF
      IF (NNB.GT.2) ID(3)=ICYCLE(ID(2),1,MXBOUND)
      IC=1
      DO 200 I=2,NNB
         IF (IEL(I).NE.-1) THEN
            IC=IC+1
            IELBND(ID(IC))=IEL(I)
            KRBND(ID(IC))=KRV(IEL(I))
            NAEXBND(1,ID(IC))=NAEX(I)
         ENDIF
  200 CONTINUE

      NNGHB=NNB
      DO 500 IB=2,NNGHB
        IDD=ICYCLE(IPBND,IB,MXBOUND)
        NBOUND(IDD)=NDIM(KRBND(IDD))-MAXDIM+1
        DO 500 I=2,4
  500   NAEXBND(I,IDD)=100
      RETURN
C----------END CONELM----------
      END


      SUBROUTINE NEIGHB(IELOLD,IELNEW,ICON,NAEXNEW)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION ICON(6)

      DO 100 I=6,1,-1
         IF (ICON(I).EQ.IELOLD) THEN
            NAEXNEW=I
            RETURN
         ENDIF
  100 CONTINUE
      NAEXNEW=0
      RETURN
C----------END NEIGHB----------
      END


      SUBROUTINE ALLELM(*)
C-------------------------------------------------
C     GET ALL ELEMENTS NEIGHBOURED AT A GIVEN EDGE OR CORNER
C-------------------------------------------------

      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
```

```
     &           ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &   ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &   ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &    XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &   ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &   ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &   ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &   ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------

      ID1=ICYCLE(IPBND,IELPOS,MXBOUND)
      IPBND=ICYCLE(ID1,-1,MXBOUND)
      IELPOS=1
      CALL GETNODE(KRBND(ID1),NAEXBND(1,ID1),NOD)

      IF (NOD.EQ.0) THEN
        WRITE (OUTRA,10000) IELM(IELBND(ID1)),KRBND(ID1)
     &   ,(XLBND(I,ID1),I=1,3),(NAEXBND(I,ID1),I=1,4),NBOUND(ID1)
        PRINT         10000, IELM(IELBND(ID1)),KRBND(ID1)
     &   ,(XLBND(I,ID1),I=1,3),(NAEXBND(I,ID1),I=1,4),NBOUND(ID1)

10000 FORMAT (' -- Warning from "ALLELM" --.The corresponding node'
     &  ,' could not be found!'
     &  /' Element No. ',I5,'  Element Type KR =',I3
     &  /' Local coordinates :',3F13.8
     &  /' Boundary Faces at this Point :',4I5
     &  /' Number of Boundary Faces     :',I3)

        NNGHB=1
        NBOUND(ID1)=0
        RETURN
      ENDIF

      NODNUM=NELC(NOD,IELBND(ID1))

      NNGHB=1
      DO 200 IEL=1,MXLM
        IF (IEL.EQ.IELBND(ID1)) GOTO 200
        DO 100 NIC=1,KRV(IEL)
          IF (NELC(NIC,IEL).EQ.NODNUM) THEN
            NNGHB=NNGHB+1
            IF (NNGHB.GT.MXBOUND) RETURN 1
            IDN=ICYCLE(IPBND,NNGHB,MXBOUND)
            KRBND(IDN)=KRV(IEL)
            IELBND(IDN)=IEL
            GOTO 200
          ENDIF
  100   CONTINUE
  200 CONTINUE
```

```
      IF (NNGHB.EQ.1) RETURN

C New neighbour elements found: store corresponding NBOUND and NAEX
      DO 500 IB=2,NNGHB
        ID=ICYCLE(IPBND,IB,MXBOUND)
        IEL=IELBND(ID)
        KR=KRBND(ID)
        DO 300 NIC=1,KR
          IF (NELC(NIC,IEL).EQ.NODNUM) THEN
            NOD=NIC
            GOTO 400
          ENDIF
  300   CONTINUE
  400   CALL GETNAEX(KRBND(ID),NOD,NBOUND(ID),NAEXBND(1,ID))
  500 CONTINUE
      RETURN
C----------END ALLELM----------
      END


      SUBROUTINE CHKOUTS(*)

      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &   ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &   ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &    XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &   ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &   ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &   ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &   ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------

      DO 100 IB=1,NNGHB
        ID=ICYCLE(IPBND,IB,MXBOUND)
        DO 100 NA=1,NBOUND(ID)
          IF (NAEXBND(NA,ID).NE.0) THEN
            IF (ICON(NAEXBND(NA,ID),IELBND(ID)).EQ.-1) RETURN 1
          ENDIF
  100 CONTINUE
      RETURN
C----------END CHKOUTS----------
      END
```

```
      SUBROUTINE AREANOD

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL EQ
      DIMENSION NAEX(4)
      DIMENSION NARNOD(4*143),KRBOUND(143),IPKR(27),NHUND(4)

      DATA NHUND /4*100/
      DATA IPKR
     &  /2*0,0,2*0,3,0,9,17,26,2*0,36,49,63,2*0,78,0,96,6*0,116/
      DATA  (NARNOD(I),I=1,12)     /  1,100,100,100
     &                              ,100,100,100,100
     &                              ,  2,100,100,100/  !* KR=3
      DATA  (NARNOD(I),I=13,36)    /  1,  3,100,100
     &                              ,  1,100,100,100
     &                              ,  1,  2,100,100
     &                              ,  2,100,100,100
     &                              ,  2,  3,100,100
     &                              ,  3,100,100,100/  !* KR=6
      DATA  (NARNOD(I),I=37,68)    /  1,  4,100,100
     &                              ,  1,100,100,100
     &                              ,  1,  2,100,100
     &                              ,  2,100,100,100
     &                              ,  2,  3,100,100
     &                              ,  3,100,100,100
     &                              ,  3,  4,100,100
     &                              ,  4,100,100,100/  !* KR=8
      DATA  (NARNOD(I),I=69,104)   /  1,  4,100,100
     &                              ,  1,100,100,100
     &                              ,  1,  2,100,100
     &                              ,  2,100,100,100
     &                              ,  2,  3,100,100
     &                              ,  3,100,100,100
     &                              ,  3,  4,100,100
     &                              ,  4,100,100,100
     &                              ,100,100,100,100/ !* KR=9
      DATA  (NARNOD(I),I=105,144)  /  1,  2,  4,100
     &                              ,  1,  2,100,100
     &                              ,  1,  2,  3,100
     &                              ,  1,  3,100,100
     &                              ,  1,  3,  4,100
     &                              ,  1,  4,100,100
     &                              ,  2,  4,100,100
     &                              ,  2,  3,100,100
     &                              ,  3,  4,100,100
     &                              ,  2,  3,  4,100/ !* KR=10
      DATA  (NARNOD(I),I=145,196)  /  1,  2,  5,100
     &                              ,  1,  2,100,100
     &                              ,  1,  2,  3,100
     &                              ,  1,  3,100,100
     &                              ,  1,  3,  4,100
     &                              ,  1,  4,100,100
     &                              ,  1,  4,  5,100
     &                              ,  1,  5,100,100
     &                              ,  2,  5,100,100
     &                              ,  2,  3,100,100
     &                              ,  3,  4,100,100
     &                              ,  4,  5,100,100
     &                              ,  2,  3,  4,  5/ !* KR=13
      DATA  (NARNOD(I),I=197,252)  /  1,  2,  5,100
     &                              ,  1,  2,100,100
     &                              ,  1,  2,  3,100
     &                              ,  1,  3,100,100
     &                              ,  1,  3,  4,100
     &                              ,  1,  4,100,100
     &                              ,  1,  4,  5,100
     &                              ,  1,  5,100,100
     &                              ,  1,100,100,100
     &                              ,  2,  5,100,100
     &                              ,  2,  3,100,100
     &                              ,  3,  4,100,100
     &                              ,  4,  5,100,100
     &                              ,  2,  3,  4,  5/ !* KR=14
      DATA  (NARNOD(I),I=253,312)  /  1,  3,  5,100
     &                              ,  1,  3,100,100
```

```
&                                ,  1,  3,  4,100
&                                ,  1,  4,100,100
&                                ,  1,  4,  5,100
&                                ,  1,  5,100,100
&                                ,  3,  5,100,100
&                                ,  3,  4,100,100
&                                ,  4,  5,100,100
&                                ,  2,  3,  5,100
&                                ,  2,  3,100,100
&                                ,  2,  3,  4,100
&                                ,  2,  4,100,100
&                                ,  2,  4,  5,100
&                                ,  2,  5,100,100/ !* KR=15
 DATA  (NARNOD(I),I=313,384) /  1,  3,  5,100
&                                ,  1,  3,100,100
&                                ,  1,  3,  4,100
&                                ,  1,  4,100,100
&                                ,  1,  4,  5,100
&                                ,  1,  5,100,100
&                                ,  3,  5,100,100
&                                ,  3,100,100,100
&                                ,  3,  4,100,100
&                                ,  4,100,100,100
&                                ,  4,  5,100,100
&                                ,  5,100,100,100
&                                ,  2,  3,  5,100
&                                ,  2,  3,100,100
&                                ,  2,  3,  4,100
&                                ,  2,  4,100,100
&                                ,  2,  4,  5,100
&                                ,  2,  5,100,100/ !* KR=18
 DATA  (NARNOD(I),I=385,464) /  2,  4,  6,100
&                                ,  4,  6,100,100
&                                ,  1,  4,  6,100
&                                ,  1,  6,100,100
&                                ,  1,  3,  6,100
&                                ,  3,  6,100,100
&                                ,  2,  3,  6,100
&                                ,  2,  6,100,100
&                                ,  2,  4,100,100
&                                ,  1,  4,100,100
&                                ,  1,  3,100,100
&                                ,  2,  3,100,100
&                                ,  2,  4,  5,100
&                                ,  4,  5,100,100
&                                ,  1,  4,  5,100
&                                ,  1,  5,100,100
&                                ,  1,  3,  5,100
&                                ,  3,  5,100,100
&                                ,  2,  3,  5,100
&                                ,  2,  5,100,100/ !* KR=20
 DATA  (NARNOD(I),I=465,524) /  2,  4,  6,100
&                                ,  4,  6,100,100
&                                ,  1,  4,  6,100
&                                ,  1,  6,100,100
&                                ,  1,  3,  6,100
&                                ,  3,  6,100,100
&                                ,  2,  3,  6,100
&                                ,  2,  6,100,100
&                                ,  6,100,100,100
&                                ,  2,  4,100,100
&                                ,  4,100,100,100
&                                ,  1,  4,100,100
&                                ,  1,100,100,100
&                                ,  1,  3,100,100
&                                ,  3,100,100,100/
 DATA  (NARNOD(I),I=525,572) /  2,  3,100,100
&                                ,  2,100,100,100
&                               ,100,100,100,100
&                                ,  2,  4,  5,100
&                                ,  4,  5,100,100
&                                ,  1,  4,  5,100
&                                ,  1,  5,100,100
&                                ,  1,  3,  5,100
&                                ,  3,  5,100,100
```

```
     &                                ,  2,  3,  5,100
     &                                ,  2,  5,100,100
     &                                ,  5,100,100,100/ !* KR=27


      DATA  (KRBOUND(I),I=1,3)      /1,0,1/
     &     ,(KRBOUND(I),I=4,9)      /2,1,2,1,2,1/
     &     ,(KRBOUND(I),I=10,17)    /2,1,2,1,2,1,2,1/
     &     ,(KRBOUND(I),I=18,26)    /2,1,2,1,2,1,2,1,0/
     &     ,(KRBOUND(I),I=27,36)    /3,2,3,2,3,2,2,2,2,3/
     &     ,(KRBOUND(I),I=37,49)    /3,2,3,2,3,2,3,2,2,2,2,2,2,4/
     &     ,(KRBOUND(I),I=50,63)    /3,2,3,2,3,2,3,2,1,2,2,2,2,4/
     &     ,(KRBOUND(I),I=64,78)    /3,2,3,2,3,2,2,2,2,3,2,3,2,3,2/
     &     ,(KRBOUND(I),I=79,96)    /3,2,3,2,3,2,2,1,2,1,2,1,3,2,3,2,3,2/
     &     ,(KRBOUND(I),I=97,116)   /3,2,3,2,3,2,3,2,2,2,2,2
     &                              ,3,2,3,2,3,2,3,2/
     &     ,(KRBOUND(I),I=117,143) /3,2,3,2,3,2,3,2,1,2,1,2,1,2,1,2,1,0
     &                              ,3,2,3,2,3,2,3,2,1/

      RETURN

      ENTRY GETNODE(KR,NAEX,NOD)

C Find node defined by given combination of the values
C NAEX(1)...NAEX(4). This node uniquely describes the corresponding
C edge or corner of the element.
      DO 100 ND=1,KR
         IP=4*(IPKR(KR)+(ND-1))
         DO 50 I=1,4
50       IF (NARNOD(IP+I).NE.NAEX(I)) GOTO 100
         NOD=ND
         RETURN
100   CONTINUE
      NOD=0
      RETURN

      ENTRY GETNAEX(KR,NOD,NBOUND,NAEX)

C Find combination of NAEX(1)...NAEX(4) for a given node.
      NBOUND=KRBOUND(IPKR(KR)+NOD)
      IF (NBOUND.GT.0)
     &   CALL IVEQV(NBOUND,NAEX,NARNOD(4*(IPKR(KR)+(NOD-1))+1))
      IF (NBOUND.LT.4)
     &   CALL IVEQV(4-NBOUND,NAEX(NBOUND+1),NHUND)
      RETURN

C----------END AREANOD----------
      END


      SUBROUTINE NXTCOOR(*,IEL,ID)

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL INS

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
C-----COMMON BOUND_COM----------
```

```
      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &   ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &   ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &    XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &   ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &   ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &   ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &   ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------

      PARAMETER (EPSG=-1D0)

      DIMENSION XL0(3),NAEX(1)

      DO 1000 IB=1,NNGHB
        IF (IB.EQ.IELPOS) GOTO 1000
        ID=ICYCLE(IPBND,IB,MXBOUND)
        IEL=IELBND(ID)
        KR=KRV(IEL)
        CALL GETXKR(KR,COOR,NELC(1,IEL),XKRBND(1,1,ID))

C get exact values for corners
        IF (BNDDIM.EQ.0) THEN
          CALL GETNODE(KR,NAEXBND(1,ID),NCORN)
          IF (NCORN.EQ.0) RETURN 1
          CALL XLOCAL(KR,NCORN,XLBND(1,ID))
          GOTO 1000
        ENDIF

        CALL LSTART(KR,XL0)
        CALL SEARCH(XGBND,XLBND(1,ID),XL0,KR,XKRBND(1,1,ID)
     &             ,SIGN(MAXDIM,NARV(IEL)),EPSG,IFAIL)
        IF(IFAIL.NE.0) THEN
           IF(IFAIL.NE.50) RETURN 1
           CALL WRERR(IELBND(ID),XLBND(1,ID),KR,NDIM(KR),'NXTCOOR')
           IFAIL=0
        ENDIF
        CALL IOTEST(XL0,XLBND(1,ID),XLBND(1,ID),KR,NAEX
     &   ,NB,INS,.TRUE.,.TRUE.,.FALSE.,.FALSE.,.FALSE.,0,0,IOERR)
 1000 CONTINUE

      RETURN
C----------END NXTCOOR----------
      END



      SUBROUTINE XLOCAL(KR,NODE,XL)
C-----------------------------------------
C   get local coordinates for specified node
C-----------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)

      PARAMETER (U=1D0,H=5D-1,Z=0D0,V=-1D0)


      DIMENSION XL(3)
      DIMENSION XLOC3(3,3),XLOC6(3,6),XLOC8(3,8),XLOC9(3,9),
```

```
     &          XLOC10(3,10),XLOC13(3,13),XLOC15(3,15),
     &          XLOC18(3,18),XLOC20(3,20),XLOC27(3,27)
      DIMENSION IPNOD(27),XLTOTAL(3,129)

      EQUIVALENCE (XLTOTAL(1,1),XLOC3)
     &           ,(XLTOTAL(1,4),XLOC6)
     &           ,(XLTOTAL(1,10),XLOC8)
     &           ,(XLTOTAL(1,18),XLOC9)
     &           ,(XLTOTAL(1,27),XLOC10)
     &           ,(XLTOTAL(1,37),XLOC13)
     &           ,(XLTOTAL(1,50),XLOC15)
     &           ,(XLTOTAL(1,65),XLOC18)
     &           ,(XLTOTAL(1,83),XLOC20)
     &           ,(XLTOTAL(1,103),XLOC27)

      DATA IPNOD
     &  /5*0,3,0,9,17,26,2*0,36,0,49,0,0,64,0,82,6*0,102/

      DATA XLOC3 / V, Z, Z,   Z, Z, Z,   U, Z, Z/
      DATA XLOC6 / U, Z, Z,   H, H, Z,   Z, U, Z,
     &             Z, H, Z,   Z, Z, Z,   H, Z, Z/
      DATA XLOC8 / V, V, Z,   Z, V, Z,   U, V, Z,  U, Z, Z,
     &             U, U, Z,   Z, U, Z,   V, U, Z,  V, Z, Z/
      DATA XLOC9 / V, V, Z,   Z, V, Z,   U, V, Z,  U, Z, Z,
     &             U, U, Z,   Z, U, Z,   V, U, Z,  V, Z, Z,
     &             Z, Z, Z/
      DATA XLOC10/ U, Z, Z,   H, H, Z,   Z, U, Z,
     &             Z, H, Z,   Z, Z, Z,   H, Z, Z,
     &             H, Z, H,   Z, H, H,   Z, Z, H,
     &             Z, Z, U/
      DATA XLOC13/ V, V, V,   Z, V, V,   U, V, V,  U, Z, V,
     &             U, U, V,   Z, U, V,   V, U, V,  V, Z, V,
     &             V, V, Z,   U, V, Z,   U, U, Z,  V, U, Z,
     &             Z, Z, U/
      DATA XLOC15/ U, Z, V,   H, H, V,   Z, U, V,
     &             Z, H, V,   Z, Z, V,   H, Z, V,
     &             U, Z, Z,              Z, U, Z,
     &                        Z, Z, Z,
     &             U, Z, U,   H, H, U,   Z, U, U,
     &             Z, H, U,   Z, Z, U,   H, Z, U/
      DATA XLOC18/ U, Z, V,   H, H, V,   Z, U, V,
     &             Z, H, V,   Z, Z, V,   H, Z, V,
     &             U, Z, Z,   H, H, Z,   Z, U, Z,
     &             Z, H, Z,   Z, Z, Z,   H, Z, Z,
     &             U, Z, U,   H, H, U,   Z, U, U,
     &             Z, H, U,   Z, Z, U,   H, Z, U/
      DATA XLOC20/ V, V, V,   Z, V, V,   U, V, V,  U, Z, V,
     &             U, U, V,   Z, U, V,   V, U, V,  V, Z, V,
     &             V, V, Z,              U, V, Z,
     &             U, U, Z,              V, U, Z,
     &             V, V, U,   Z, V, U,   U, V, U,  U, Z, U,
     &             U, U, U,   Z, U, U,   V, U, U,  V, Z, U/
      DATA XLOC27/ V, V, V,   Z, V, V,   U, V, V,  U, Z, V,
     &             U, U, V,   Z, U, V,   V, U, V,  V, Z, V,
     &             Z, Z, V,
     &             V, V, Z,   Z, V, Z,   U, V, Z,  U, Z, Z,
     &             U, U, Z,   Z, U, Z,   V, U, Z,  V, Z, Z,
     &             Z, Z, Z,
     &             V, V, U,   Z, V, U,   U, V, U,  U, Z, U,
     &             U, U, U,   Z, U, U,   V, U, U,  V, Z, U,
     &             Z, Z, U/

      IP=IPNOD(KR)+NODE
      DO 100 I=1,3
  100 XL(I)=XLTOTAL(I,IP)

      RETURN
C-----------END XLOCAL-----------
      END
```

```
      SUBROUTINE NXTVELO(IEL,ID)

      IMPLICIT REAL*8 (A-H,O-Z)
C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON BOUND_COM----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &   ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &   ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &    XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &   ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &   ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &   ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &   ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------

      REAL*8 NULLV(3)
      DATA NULLV/3*0D0/

      DO 1000 IB=1,NNGHB
        IF (IB.EQ.IELPOS) GOTO 1000
        ID=ICYCLE(IPBND,IB,MXBOUND)
        IEL=IELBND(ID)
        KR=KRBND(ID)
        PORBND(ID)=PORV(NELP(IEL))
        CALL GETPOT(KR,NELC(1,IEL),POT,HEADBND(1,ID))
        CALL GETPERM(PERA(1,NELP(IEL)),PERMBND(1,1,ID))

        CALL VELOTXT(IEL,XLBND(1,ID),XKRBND(1,1,ID),PERMBND(1,1,ID)
     &    ,PORBND(ID),HEADBND(1,ID),KR,NDIM(KR)
     &    ,QLBND(1,ID),QGBND(1,ID),FMBND(1,ID),FMIBND(1,ID)
     &    ,IFAIL,'NXTVELO')
        IF(IFAIL.NE.0) THEN
          CALL VEQV(3,QGBND(1,ID),NULLV)
          CALL VEQV(3,QLBND(1,ID),NULLV)
        ENDIF
 1000 CONTINUE
      RETURN
C----------END NXTVELO----------
      END
```

```
      SUBROUTINE VELOTST(*)

      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &   ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &   ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &    XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &   ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &   ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &   ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &   ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------
C-----COMMON STEP_COM-----------

      PARAMETER (MAXITER=100,DFSTEP=1D-1,SUBRA=5D-4
     &          ,SUBSTEP=2D0,NCHECK=10,DFDSCR=1D-1
     &          ,DEFDIST=1D-2)

      LOGICAL   BNDMAX,KUTTA
      COMMON /STEP/ NITTOT,TIMETOT,DISTTOT,NITELM,TIMEELM,DISTELM,DISTO
     &             ,XGSTEP(3),DXSTEP,DSSTEP,DISCRQ,NCHK,BNDMAX,KUTTA
     &             ,DISTMNQ,XGODIST(3)
C----------END STEP_COM---------

      LOGICAL ANDV,ORV,VOUT1(4),VOUT2(4)
      DIMENSION QLP(3,4),QGP(3,4),NAEXPP(4)

      MXFLOW=0
      MINDIM=3

C Loop over all neighboured elements
      DO 1000 IB=1,NNGHB
        ID=ICYCLE(IPBND,IB,MXBOUND)

C Initialize
        IQMAX=1
        KR=KRBND(ID)
        N=NDIM(KR)
        NB=NBOUND(ID)
        MINDIM=MIN(MINDIM,N)
        NFLOWIN(ID)=N
        NAEXP(ID)=0
        BNDSINK(ID)=.FALSE.
```

```
C Check direction of flux: (into element,out of element
C -----------------------------------------------------

        CALL VELIO(KR,MXNB,NB,NAEXBND(1,ID),QLBND(1,ID),VOUT1)
        BNDFLOW(ID)=ORV(MXNB,VOUT1)

C CASE 1: Flux INTO element (--> no projection)
C ---------------------------
        IF (.NOT.BNDFLOW(ID)) GOTO 500


C******************* PROJECTION ******************************

C CASE 2: Flux OUT of element (--> projection)
C ----------------------------------------

C  -- 1.check if component along NAEX(IF) points inside
C       ---------------------------------------------
        NFLOWIN(ID)=MAX(N-1,0)
        IF (NFLOWIN(ID).LT.1) GOTO 500

        DO 100 IF=1,NB
C   skip boundary if Flux points inside
          IF (.NOT.VOUT1(IF)) GOTO 100
          CALL BNDPROJ(ID,1,IF,0,QLP,QGP,NAEXPP(1))
          CALL VELIO(KR,MXNB,NB,NAEXBND(1,ID),QLP,VOUT2)
          VOUT2(IF)=.FALSE.
          IF (.NOT.ORV(NB,VOUT2)) GOTO 500
  100   CONTINUE

C  -- 2.check if component along edge NAEX(IF),NAEX(JF) points inside
C       ------------------------------------------------------------

        NFLOWIN(ID)=MAX(N-2,0)
        IF (NFLOWIN(ID).LT.1) GOTO 500

C     get edge and projection
        ICIN=0
        DO 300 IF=1,NB
          DO 200 JF=IF+1,NB
            IF (.NOT.(VOUT1(IF).OR.VOUT1(JF))) GOTO 200
            IF (NB.EQ.4.AND.(JF-IF).EQ.2) GOTO 200 !* Pyramid handling

            IC=ICIN+1
            CALL BNDPROJ(ID,2,IF,JF,QLP(1,IC),QGP(1,IC),NAEXPP(IC))
            CALL VELIO(KR,MXNB,NB,NAEXBND(1,ID),QLP(1,IC),VOUT2)
            VOUT2(IF)=.FALSE.
            VOUT2(JF)=.FALSE.
            IF (.NOT.ORV(NB,VOUT2))
     &        ICIN=ICIN+1 !* counts edges with inflow
  200     CONTINUE
  300   CONTINUE

C Select edge with maximum flux as flow boundary
        IF (ICIN.EQ.0) THEN
          NFLOWIN(ID)=0
          GOTO 500
        ELSEIF (ICIN.EQ.1) THEN
          GOTO 500
        ENDIF

        QGMAX=-1D0
        DO 400 I=1,ICIN
          QGN=MAX(VSCP(N,QGP(1,I),QGP(1,I)),QGMAX)
          IF (QGN.GT.QGMAX) THEN
            QGMAX=QGN
            IQMAX=I
          ENDIF
  400   CONTINUE
C*************************************************************

  500   CONTINUE
        IF (BNDFLOW(ID).AND.NFLOWIN(ID).GT.0) THEN
          NAEXP(ID)=NAEXPP(IQMAX)
```

```
          CALL VEQV(N,QLBND(1,ID),QLP(1,IQMAX))
          CALL VEQV(N,QGBND(1,ID),QGP(1,IQMAX))
        ENDIF

C get next velocity and check if sink encountered
        IF (NFLOWIN(ID).GT.0) THEN
          CALL BNDVEL2(ID)
          CALL CHKSINK(ID)
          BNDSINK(ID)=.FALSE. !* BNDVEL2 not yet correct,but result is
                              !*   for debugging purpose
        ENDIF

        MXFLOW=MAX(MXFLOW,NFLOWIN(ID))

 1000 CONTINUE

      IF (MXFLOW.EQ.0) RETURN 1
      NFLCRIT=1
      IF(MINDIM.EQ.MAXDIM .AND. MXFLOW.EQ.MAXDIM .AND. .NOT.BNDMAX)
     &  NFLCRIT=MAXDIM

      NELMXFL=0
      DO 1100 IB=1,NNGHB
        ID=ICYCLE(IPBND,IB,MXBOUND)
        FLOWIN(ID)=(NFLOWIN(ID).GE.NFLCRIT.AND..NOT.BNDSINK(ID))
        IF (FLOWIN(ID)) NELMXFL=NELMXFL+1
 1100 CONTINUE
      IF (NELMXFL.EQ.0) RETURN 1

      RETURN
C----------END VELOTST----------
      END



      SUBROUTINE BNDVEL2(ID)
      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &   ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &   ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &    XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &   ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &   ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &   ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
```

```
     &    ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &               , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------
C-----COMMON STEP_COM-----------

      PARAMETER (MAXITER=100,DFSTEP=1D-1,SUBRA=5D-4
     &          ,SUBSTEP=2D0,NCHECK=10,DFDSCR=1D-1
     &          ,DEFDIST=1D-2)

      LOGICAL   BNDMAX,KUTTA
      COMMON /STEP/ NITTOT,TIMETOT,DISTTOT,NITELM,TIMEELM,DISTELM,DISTO
     &             ,XGSTEP(3),DXSTEP,DSSTEP,DISCRQ,NCHK,BNDMAX,KUTTA
     &             ,DISTMNQ,XGODIST(3)
C----------END STEP_COM---------

      REAL*8 NULLV(3)
      DATA NULLV /3*0D0/

      IEL=IELBND(ID)
      KR=KRBND(ID)
      N=NDIM(KR)
      DSSTEPL=STSIZE(KR)
      IF (BNDFLOW(ID)) DSSTEPL=DSSTEPL/SUBSTEP
      QLABS=VABS(N,QLBND(1,ID))

      IF (QLABS.LE.0D0) GOTO 9999
      DT=DSSTEPL/QLABS

      DO 100 I=1,N
  100 XL2BND(I,ID)=XLBND(I,ID)+DT*QLBND(I,ID)

      CALL VELOX(XL2BND,XKRBND(1,1,ID),PERMBND(1,1,ID)
     & ,PORBND(ID),HEADBND(1,ID),KR,N,QL2BND(1,ID),QG2BND(1,ID)
     & ,FMBND(1,ID),FMIBND(1,ID),IERR)
      IF (IERR.NE.0.AND.IERR.NE.50) GOTO 9999

C projection if previous velocity was projected
C  should be done

      RETURN

C  Errors
 9999 CONTINUE
      CALL VEQV(3,QG2BND(1,ID),NULLV)
      CALL VEQV(N,QL2BND(1,ID),NULLV)
      RETURN
C----------END BNDVEL2----------
      END


      SUBROUTINE BNDPROJ(ID,NPROJ,IDA1,IDA2,VLP,VGP,NAEXPP)

      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON BOUND_COM-----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
```

```
     &     BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &    ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &    ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &     XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &    ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &    ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &    ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &    ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------

      DIMENSION VLP(3),VGP(3),VLPP(3),VGPP(3)

C-------------------------------------------------------------
C  Gets projection of local and global velocity vectors along
C    1- and 2-D boundaries
C
C    NPROJ  : Flow Dimension decrementation (input)
C    ID     : current BOUND_COM Element     (input)
C    IDA1   : 1. Index of NAEXBND           (input)
C    IDA2   : 2. Index of NAEXBND           (input)
C    VLP    : projection of local vector    (output)
C    VGP    : projection of global vector   (output)
C    NAEXPP : unique combination of IDA1 and IDA2 (output)
C-------------------------------------------------------------

      IF (NPROJ.EQ.1) THEN
        NAEXPP=NAEXBND(IDA1,ID)
      ELSEIF (NPROJ.EQ.2) THEN
        NAEXPP=10*MIN(NAEXBND(IDA1,ID),NAEXBND(IDA2,ID))
     &            +MAX(NAEXBND(IDA1,ID),NAEXBND(IDA2,ID))
      ELSE
        NAEXPP=0
        RETURN
      ENDIF

      KR=KRBND(ID)
      N=NDIM(KR)

C get projection boundary type (face->2,edge->1)
      NPTYPE=N-NPROJ

C  Projection onto face (only 3d elements)
      IF (NPTYPE.EQ.2) THEN

C   -- get local normal to NAEXPP (--> global)
        CALL VLNORM(KR,NAEXPP,VLPP)
        CALL MMAB(3,3,1,FMBND(1,ID),VLPP,VGPP) !* Transformation
                                               !*  of covariant vector

C   -- get global projection (-->local);suppress projected component
        CALL VPROJ(3,VGPP,QGBND(1,ID),VGPP)
        CALL VDIF(3,QGBND(1,ID),VGPP,VGP)
        CALL MMATB(3,3,1,FMBND(1,ID),VGP,VLP)
        CALL PROJECT(1,KR,NAEXPP,VLP,VLP)

C  Projection onto edge (2 & 3d elements)
      ELSEIF (NPTYPE.EQ.1) THEN

C   -- get local tangential along NAEXPP (--> global)
        CALL VLTANG(KR,NAEXPP,VLPP)
        CALL MMATB(3,N,1,FMIBND(1,ID),VLPP,VGPP) !* Transformation
                                                 !*  of contravariant vector

C   -- get global projection (-->local);suppress projected component
        CALL VPROJ(3,VGPP,QGBND(1,ID),VGP)
        CALL MMATB(N,3,1,FMBND(1,ID),VGP,VLP)
        CALL PROJECT(NPROJ,KR,NAEXPP,VLP,VLP)
      ENDIF

      RETURN
C----------END BNDPROJ----------
      END
```

```
      SUBROUTINE CHKSINK(ID)

      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON BOUND_COM----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &   ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &   ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &    XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &   ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &   ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &   ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &   ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------

      BNDSINK(ID)=(VSCP(3,QG2BND(1,ID),QGBND(1,ID)).LE.0D0)
      RETURN
C----------END CHKSINK----------
      END


      SUBROUTINE VELIO(KR,MXNB,NBOUND,NAEX,QL,VOUT)
C-------------------------------------------------------
C   Checks if local vector  QL  is pointing out of boundary
C   faces defined by NAEX(1..NBOUND)
C
C   VOUT(1..NBOUND)  is .TRUE. for outpointing
C   VOUT(NBOUND+1..MXNB) is always reset to .FALSE.
C-------------------------------------------------------

      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 L(6),LQ(6),DL(6),LO(6)

      LOGICAL VOUT
      DIMENSION NAEX(MXNB),VOUT(MXNB),QL(3),QO(3),QLU(3)

      DATA QO /3*0D0/
      CALL VUNIT(3,QL,QLU)
      CALL VOLCOR(KR,QO,LO,.TRUE.)
      CALL VOLCOR(KR,QLU,L,.TRUE.)
      CALL VDIF(NUMAREA(KR),L,LO,DL)

      DO 100 I=1,NBOUND
  100 VOUT(I)=(DL(NAEX(I)).LT.0D0)
      DO 200 I=NBOUND+1,MXNB
  200 VOUT(I)=.FALSE.
      RETURN
C----------END VELIO----------
      END


      SUBROUTINE MAXVELO

      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON BLANK_COM-------
```

```
      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
     &          ,XORBLK(3)
     &          ,NELP(IV2),KRV(IV2),NARV(IV2)
     &          ,PERA(6,IV3),PORV(IV3)
     &          ,MXLM,MNNIC,MXNIC,MAXDIM
     &          ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON BOUND_COM----------

      PARAMETER (MXBOUND=40,MXNB=4)

      LOGICAL BNDFLOW,LOWDIM,ONESTEP,VELOUT,OUTOFEL,CHKBND,FLOWIN
     &       ,NEWDIM,BNDSINK

      INTEGER FLOWDIM,BNDDIM

      COMMON /IBOUND/
     &    FLOWDIM,NELMXFL,NFLOWIN(MXBOUND),NBOUND(MXBOUND),NNGHB
     &   ,IPBND,IELPOS,IELBND(MXBOUND),NAEXBND(MXNB,MXBOUND),BNDDIM
     &   ,KRBND(MXBOUND),NAEXP(MXBOUND)

      COMMON /LBOUND/
     &    BNDFLOW(MXBOUND),FLOWIN(MXBOUND)
     &   ,CHKBND,NEWDIM,LOWDIM,ONESTEP,OUTOFEL
     &   ,BNDSINK(MXBOUND)

      COMMON /RBOUND/
     &    XGBND(3),XLBND(3,MXBOUND),QGBND(3,MXBOUND)
     &   ,QLBND(3,MXBOUND),FMBND(9,MXBOUND),FMIBND(9,MXBOUND)
     &   ,XKRBND(3,27,MXBOUND),PERMBND(3,3,MXBOUND),PORBND(MXBOUND)
     &   ,HEADBND(27,MXBOUND),VLNBND(3,MXBOUND),VLTBND(3,MXBOUND)
     &   ,QL2BND(3,MXBOUND),QG2BND(3,MXBOUND),XL2BND(3,MXBOUND)

C----------END BOUND_COM--------

      IF (NELMXFL.EQ.1) THEN
        DO 100 IB=NNGHB,1,-1
          ID=ICYCLE(IPBND,IB,MXBOUND)
          IF (FLOWIN(ID)) THEN
            IELPOS=IB
            GOTO 300
          ENDIF
  100   CONTINUE
      ELSE
        ID=ICYCLE(IPBND,IELPOS,MXBOUND)
        IF (FLOWIN(ID)) THEN
          QGQ=VSCP(3,QGBND(1,ID),QGBND(1,ID))
          IQMAX=IELPOS
        ELSE
          QGQ=-1D0
        ENDIF
        DO 200 IB=1,NNGHB
          IF (IB.EQ.IELPOS) GOTO 200
          ID=ICYCLE(IPBND,IB,MXBOUND)
          IF (FLOWIN(ID)) THEN
            QGQN=VSCP(3,QGBND(1,ID),QGBND(1,ID))
            IF (QGQN.GT.QGQ) THEN
              QGQ=QGQN
              IQMAX=IB
            ENDIF
          ENDIF
  200   CONTINUE
        IELPOS=IQMAX
      ENDIF

  300 CONTINUE
      ID=ICYCLE(IPBND,IELPOS,MXBOUND)
      N=NDIM(KRBND(ID))
      NFL=NFLOWIN(ID)
      NEWDIM=(NFL.NE.FLOWDIM)
```

```
      FLOWDIM=NFL
      ONESTEP=(FLOWDIM.LT.MAXDIM)
      CHKBND=((FLOWDIM.GT.BNDDIM).AND.ONESTEP)
      LOWDIM=(N.LT.MAXDIM)
      IF ((.NOT.ONESTEP).OR.CHKBND) THEN
        IPBND=ICYCLE(IPBND,IELPOS-1,MXBOUND)
        IELPOS=1
      ENDIF
      RETURN
C----------END MAXVELO----------
      END


      SUBROUTINE SEARCH(XGIN,XL,XLIN,KR,XKR,MDPCH,EPSILON,ICTR)
C-------------------------------------------------------------------------
C      SEARCH CALCULATES THE LOCAL COORDINATES GIVEN A POINT IN
C      GLOBAL COORDINATES.
C      THE ITERATION PROCESS IS ANALOGOUS TO THE NEWTON-APPROXIMATION-
C      METHOD IN 1 DIMENSION.
C
C      VERSION 1.1  02-JUNE-1988  AVK
C      02-JUNE-1988 : PARAMETER LIST HAS CHANGED;MDPCH IS PASSED
C                     MDPCH : signed model dimensionality
C                      (<0 for pinched elements)
C
C-------------------------------------------------------------------------
C-------------------------------------------------------------------------
C  ICTR  : CONTROL-PARAMETER
C    =0  : CONVERGENCE ACHIEVED
C   =10  : DET=0.
C   =20  : CONVERGENCE NOT ACHIEVED (LOCAL COORDINATES)
C   =30  : POINT LIES OUTSIDE THE ELEMENT
C   =50  : DET<0.
C   =60  : CONVERGENCE NOT ACHIEVED (GLOBAL COORDINATES)
C
C-------------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)

      LOGICAL INS,T,F,NEGDET,PINCH,LFOUND,GFOUND
      PARAMETER (ERR=1D-6,ERRLQ=ERR**2,ERRGQ=1D-5
     &          ,ERRLOW=5D0,T=.TRUE.,F=.FALSE.)
      DIMENSION XGIN(3),XLIN(*),XL(*),XG(3),FD(81)
     &         ,XKR(3,*),DXG(3),DXL(3),SFM(9),XLN(3),XLE(3),XLO(3)
     &         ,SFMINV(9),NAEX(4)

C--------------  INITIAL VALUES ---------------------------------------

C  Defining error values
      ERRG=SQRT(ERRGQ)
      EPSLQ=ERRLQ
      IF (EPSILON.GT.0D0) THEN
        EPSG=EPSILON
        EPSGQ=EPSILON**2
      ELSE
        EPSG=ERRG
        EPSGQ=ERRGQ
      ENDIF

C  get local coordinates directly for 1d elements
      IF (KR.EQ.3) THEN
        CALL SEARCH1(XGIN,XL,XKR,EPSG,ICTR)
        RETURN
      ENDIF

      N=NDIM(KR)
      CALL VEQV(N,XL,XLIN)
      CALL LSTART(KR,XLO)
      ICTR=0
      PINCH=(MDPCH.LT.0)
      LFOUND=.FALSE.
      GFOUND=.FALSE.
C---------------------ITERATION STARTS--------------------------------
      DO 1000 ITER=1,20
```

```
C------------ GET SFM (ANALOGOUS TO THE INVERSE OF JACOBIAN) -----------
         CALL METRIC(XKR,XL,N,KR,SFM,SFMINV,FD,ICTR)
         NEGDET=(ICTR.EQ.50)
         CALL IOTEST(XLIN,XL,XLE,KR,NAEX,NB,INS,F,F,F,F,F,0,0,IERR)
         IF (INS) CALL VEQV(N,XLO,XL)
C --- Following test is skipped,but be very careful
C          IF (ICTR.NE.0) THEN
C            CALL IOTEST(XLO,XL,XL,KR,NAEX,NB,INS,F,T,F,F,F,0,0,IERR)
C            IF (.NOT.INS .OR. NEGDET) GOTO 1000
C            GOTO 2000
C          ENDIF

C--------- CALCULATING G.C. OF XL AND DIFFERENCE TO THE INITIAL POINT---
         CALL SHAPEV(XL,XG,XKR,KR)
         CALL VDIF(3,XG,XGIN,DXG)

C----------------DIFFERENCE IN L.C. (=DXL) AND IN G.C. (=DXG)-----------
         CALL MMATB(N,3,1,SFM,DXG,DXL)
         CALL VDIF(N,XL,DXL,XLN)
         IF (.NOT.PINCH) THEN
            IF (VSCP(N,DXL,DXL).GT.EPSLQ) GOTO 500
            LFOUND=.TRUE.
         ENDIF
         IF (VSCP(3,DXG,DXG).LT.EPSG) THEN
           CALL VEQV(N,XL,XLN)
           GFOUND=.TRUE.
           GOTO 2000
         ENDIF
  500    CALL VEQV(N,XL,XLN)

 1000 CONTINUE
      IF (.NOT.LFOUND) THEN
        ICTR=20    !* no convergence in local coordinates
      ELSEIF (.NOT.GFOUND) THEN
        ICTR=60    !* no convergence in global coordinates
      ENDIF
      RETURN

 2000 IF(NEGDET) ICTR=50
      RETURN

C----------END SEARCH-----------
      END


      SUBROUTINE SEARCH1(XG,XL,XKR,EPSILON,ICTR)
C-----------------------------------------------------------------------
C     find local coordinates for 1d elements
C      solving the quadratic equation 0=A(x*x)+Bx+C
C
C     ICTR  =   0 : XG found
C     ICTR  =  30 : XG not found
C
C     EPSILON     : error distance
C-----------------------------------------------------------------------

      IMPLICIT REAL*8 (A-H,O-Z)

      PARAMETER (Z=0D0)

      LOGICAL SOL(2)

      DIMENSION XG(3),XL(1),XKR(3,3),XLS(2),XGG(3),XLL(3)
     &          ,DXG(3)

      DATA XLL/3*Z/

      EPS=MAX(EPSILON,Z)
      EPSQ=EPS**2
      BND=1D0+EPS
      SOL(1)=.FALSE.
      SOL(2)=.FALSE.
```

```
      B=5D-1*(-XKR(1,1)+XKR(1,3))
      A=5D-1*( XKR(1,1)+XKR(1,3))-XKR(1,2)
      C=XKR(1,2)-XG(1)

C Distinguish constant,linear,quadratic cases
      IF (.NOT.(ABS(A).GT.EPS)) THEN

        IF (.NOT.(ABS(B).GT.EPS)) THEN
C     constant case
          IF (ABS(C).GT.EPS) GOTO 3000
          XLS(1)=1D0
        ELSE
C     linear case
          XLS(1)=-C/B
          IF (ABS(XLS(1)).GT.BND) GOTO 3000
        ENDIF

        SOL(1)=.TRUE.
        NSOL=1

      ELSE

C     quadratic case
        A2=B/A/2D0
        B2=C/A
        DQ=A2**2-B2
        IF (DQ.LT.Z) GOTO 3000
        D=SQRT(DQ)
        NSOL=0
        DO 100 I=1,2
          J=2*I-3
          XLS(I)=-(A2+J*D)
          IF (ABS(XLS(I)).GT.BND) GOTO 100
          SOL(I)=.TRUE.
          NSOL=NSOL+1
  100   CONTINUE
        IF (NSOL.LT.1) GOTO 3000
      ENDIF

C  Check if local point fulfills shape functions
      DO 200 I=1,2
        IF (.NOT.SOL(I)) GOTO 200
        XLL(1)=XLS(I)
        CALL SHAPEV(XLL,XGG,XKR,3)
        CALL VDIF(3,XGG,XG,DXG)
        IF (VSCP(3,DXG,DXG).GT.EPSQ) GOTO 200

C   shape function okay
        XL(1)=XLL(1)
        ICTR=0
        RETURN

  200 CONTINUE

C   no solution found
 3000 CONTINUE
      ICTR=30
      RETURN
C----------END SEARCH1----------
      END



      SUBROUTINE WRERR(IEL,XL,KR,N,SUBR)
C---------------------------------------------------------------------
C     PUTS ERROR MESSAGE TO FILE OUTRA AND ONTO SCREEN
C---------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)

C-----COMMON BLANK_COM-------

      PARAMETER (IV1=100000,IV2=10000,IV3=1000)

      COMMON //  IDCOR(IV1),COOR(3,IV1),POT(IV1)
     &          ,NELC(27,IV2),IELM(IV2),INVELM(IV2),ICON(6,IV2)
```

```
     &           ,XORBLK(3)
     &           ,NELP(IV2),KRV(IV2),NARV(IV2)
     &           ,PERA(6,IV3),PORV(IV3)
     &           ,MXLM,MNNIC,MXNIC,MAXDIM
     &           ,IELV(IV2)

C----------END BLANK_COM--------
C-----COMMON TAPES_COM-------
      INTEGER OUPAT,OUTRA,OUATD,OULVT,OUTAB
      COMMON /TAPES/  INELM,INCOR,INPAR,INPOT,INSTA,OUPAT,OUTRA,IOMAT
     &              , OUATD,OULVT,OUTAB
C----------END TAPES_COM--------

      CHARACTER*(*) SUBR,FMT*30,TXT(5)*133
      DIMENSION LTXT(5)
      DIMENSION XL(N)

      CALL LENWOB(SUBR,LSUBR)

      FMT='(A,A,A)'
      WRITE (TXT(1),FMT)
     &  'Warning (',SUBR(:LSUBR),'):DET < 0 in routine METRIC'

      FMT='(A,I5)'
      WRITE (TXT(2),FMT)
     &  'EI=',IELM(IEL)
      FMT='(A,I3)'
      WRITE (TXT(3),FMT)
     &  ' KR=',KR
      FMT='(A,I3)'
      WRITE (TXT(4),FMT)
     &  ' NAR=',NARV(IEL)
      WRITE (FMT,'(''(A,'',I1,''F8.3,A)'')') N
      WRITE (TXT(5),FMT)
     &  ' XLBND(1...N)=',(XL(J),J=1,N)

      DO 100 J=1,5
  100 CALL LENWOB(TXT(J),LTXT(J))

      WRITE (OUTRA,20000) (TXT(J)(:LTXT(J)),J=1,5)
      WRITE (*     ,20001) (TXT(J)(:LTXT(J)),J=1,5)

      IERR=0
      RETURN

20000 FORMAT(A,' (',4A,')')
20001 FORMAT(1X,A,' (',4A,')')
C----------END WRERR-----------
      END


      SUBROUTINE VELOTXT(IEL,XL,XKR,PERM,POR,H,KR,N,QL,QG
     &                  ,FM,FMI,IERR,SUBR)
C-----------------------------------------------------------------------
C     CALLS VELOX AND WRERR
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)

      DIMENSION XL(N),XKR(3,KR),PERM(3,3),H(KR),QL(N),QG(3),FM(9)
     &         ,FMI(9),V1(3),V2(3),FD(81)

      CALL VELOX(XL,XKR,PERM,POR,H,KR,N,QL,QG,FM,FMI,IERR)
      IF (IERR.NE.50) RETURN

      CALL WRERR(IEL,XL,KR,N,'VELOX')
      IERR=0
C----------END VELOTXT----------
      END
```

```
      SUBROUTINE VELOX(XL,XKR,PERM,POR,H,KR,N,QL,QG,FM,FMI,IERR)
C-----------------------------------------------------------------------
C     VELOCITY-CALCULATION BY MATRIX-MULTIPLICATION
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XL(N),XKR(3,KR),PERM(3,3),H(KR),QL(N),QG(3),FM(9)
     &       ,FMI(9),V1(3),V2(3),FD(81),XLOC(3)

      CALL METRIC(XKR,XL,N,KR,FM,FMI,FD,IERR)
      IF (IERR.NE.0 .AND. IERR.NE.50) THEN
         DO 20 I=1,3
20       XLOC(I)=.999*XL(I)
         CALL METRIC(XKR,XLOC,N,KR,FM,FMI,FD,IERR)
         IF (IERR.NE.0 .AND. IERR.NE.50) RETURN
      ENDIF
      CALL MMATB(N,KR,1,FD,H,V1)
      CALL MMAB(3,N,1,FM,V1,V2)
      CALL MMAB(3,3,1,PERM,V2,V1)
      DO 10 I=1,3
   10 QG(I)=-V1(I)/POR
      CALL MMATB(N,3,1,FM,QG,QL)
      RETURN
C----------END VELOX-----------
      END


      SUBROUTINE RUNGE(IEL,X1,DT,QL,QG,XKR,PERM,POR,H,KR,N,IERR)

C-----------------------------------------------------------------------
C     INTEGRATION BY 4TH ORDER RUNGE KUTTA METHOD
C-----------------------------------------------------------------------

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X1(3),QL(3),FKL(3,4),XK(3),FKG(3,4)
     &         ,XKR(3,*),PERM(*),H(*),FM(9),FMINV(9)
     &         ,QG(3),W(3),XG(3)

      DATA W/0.5D0,0.5D0,1D0/

      DO 10 I=1,N
   10 FKL(I,1)=QL(I)
      DO 20 I=1,3
   20 FKG(I,1)=QG(I)
      DO 200 J=1,3
        DO 100 I=1,N
  100   XK(I)=X1(I)+DT*W(J)*FKL(I,J)
        CALL VELOX(XK,XKR,PERM,POR,H,KR,N,FKL(1,J+1),FKG(1,J+1)
     &   ,FM,FMINV,IERR)
        IF (IERR.NE.0 .AND. IERR.NE.50) RETURN
  200 CONTINUE
      DO 300 I=1,N
  300 QL(I)=(FKL(I,1)+2D0*(FKL(I,2)+FKL(I,3))+FKL(I,4))/6D0
      DO 400 I=1,3
  400 QG(I)=(FKG(I,1)+2D0*(FKG(I,2)+FKG(I,3))+FKG(I,4))/6D0

      IF (IERR.EQ.50)
     &   CALL WRERR(IEL,X1,KR,N,'RUNGE')

      RETURN
C----------END RUNGE-----------
      END


      SUBROUTINE METRIC(XKR,S,N,KR,SFM,SFMINV,FD,ICTR)

C-----------------------------------------------------------------------|
C       CALCULATION OF THE METRIC TENSOR : GKOV,GKTR                    |
C       GIVEN BY G=SFMINV(TR)*SFMINV AND SFMINV BEING THE SINGULAR      |
C       JACOBIAN                                                        |
C       SFMINV=DX/DS.                                                   |
C       CALCULATION OF SFM = SFMINV * G(-1) TO GET THE ANALOGON         |
C       TO THE  FUNCTIONAL MATRIX OF A N-N MAPPING.                     |
C-----------------------------------------------------------------------|
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XKR(3*KR),S(N),FD(KR*N),SFM(3*N),SFMINV(3*N)
     &          ,GKOV(9),GKTR(9)

      CALL JACOBI(XKR,S,KR,N,SFMINV,FD)
      IF (N.EQ.3) THEN
        CALL MINV3(3,SFMINV,SFM,DET,ICTR)
        RETURN
      ENDIF
      CALL MMABT(N,3,N,SFMINV,SFMINV,GKOV)
      CALL MINV3(N,GKOV,GKTR,DET,ICTR)
      IF (ICTR.NE.0 .AND. ICTR.NE.50) RETURN
      CALL MMATB(3,N,N,SFMINV,GKTR,SFM)
      RETURN
C----------END METRIC----------
      END


      SUBROUTINE VLNORM(KR,NAEXPP,VN)

      IMPLICIT REAL*8 (A-H,O-Z)

      PARAMETER (H=5D-1,T=1D0/3D0)

      DIMENSION VN(*)

      HH=SQRT(H)
      TH=SQRT(T)
      N=NDIM(KR)

      DO 10 I=1,N
   10 VN(I)=0D0

      IF(KR.EQ.3) THEN
         CONTINUE

      ELSEIF (KR.EQ.6) THEN
         IF(NAEXPP.EQ.1) THEN
            VN(1)=HH
            VN(2)=HH
         ELSEIF (NAEXPP.EQ.2) THEN
            VN(1)=-1D0
         ELSEIF (NAEXPP.EQ.3) THEN
            VN(2)=-1D0
         ENDIF

      ELSEIF (KR.EQ.8.OR.KR.EQ.9) THEN
         IF(NAEXPP.EQ.1.OR.NAEXPP.EQ.3) THEN
            VN(2)=NAEXPP-2D0
         ELSEIF (NAEXPP.EQ.2.OR.NAEXPP.EQ.4) THEN
            VN(1)=3D0-NAEXPP
         ENDIF

      ELSEIF (KR.EQ.10) THEN
         IF(NAEXPP.EQ.1) THEN
            VN(3)=-1D0
         ELSEIF (NAEXPP.EQ.2) THEN
            VN(1)=TH
            VN(2)=TH
            VN(3)=TH
         ELSEIF (NAEXPP.EQ.3) THEN
            VN(1)=-1D0
         ELSEIF (NAEXPP.EQ.4) THEN
            VN(2)=-1D0
         ENDIF

      ELSEIF (KR.EQ.13.OR.KR.EQ.14) THEN
         IF(NAEXPP.EQ.1) THEN
            VN(3)=-1D0
         ELSEIF (NAEXPP.EQ.2.OR.NAEXPP.EQ.4) THEN
            VN(2)=NAEXPP-3D0
         ELSEIF (NAEXPP.EQ.3.OR.NAEXPP.EQ.5) THEN
            VN(1)=4D0-NAEXPP
```

```
         ENDIF

      ELSEIF (KR.EQ.15.OR.KR.EQ.18) THEN
         IF(NAEXPP.EQ.1.OR.NAEXPP.EQ.2) THEN
            VN(3)=2D0*NAEXPP-3D0
         ELSEIF (NAEXPP.EQ.3) THEN
            VN(1)=HH
            VN(2)=HH
         ELSEIF (NAEXPP.EQ.4) THEN
            VN(1)=-1D0
         ELSEIF (NAEXPP.EQ.5) THEN
            VN(2)=-1D0
         ENDIF

      ELSEIF (KR.EQ.20.OR.KR.EQ.27) THEN
         IF(NAEXPP.EQ.1.OR.NAEXPP.EQ.2) THEN
            VN(1)=3D0-2D0*NAEXPP
         ELSEIF (NAEXPP.EQ.3.OR.NAEXPP.EQ.4) THEN
            VN(2)=7D0-2D0*NAEXPP
         ELSEIF (NAEXPP.EQ.5.OR.NAEXPP.EQ.6) THEN
            VN(3)=11D0-2D0*NAEXPP
         ENDIF

      ENDIF

      RETURN
C----------END VLNORM----------
      END


      SUBROUTINE VLTANG(KR,NA,VT)

      IMPLICIT REAL*8 (A-H,O-Z)

      PARAMETER (H=5D-1,T=1D0/3D0)

      DIMENSION VT(*)

      HH=SQRT(H)
      TH=SQRT(T)
      N=NDIM(KR)

      DO 10 I=1,N
   10 VT(I)=0D0

      IF(KR.EQ.3) THEN
         VT(1)=3D0-2D0*NA

      ELSEIF (KR.EQ.6) THEN
         IF(NA.EQ.1) THEN
            VT(1)=HH
            VT(2)=-HH
         ELSEIF (NA.EQ.2) THEN
            VT(2)=1D0
         ELSEIF (NA.EQ.3) THEN
            VT(1)=1D0
         ENDIF

      ELSEIF (KR.EQ.8.OR.KR.EQ.9) THEN
         IF(NA.EQ.1.OR.NA.EQ.3) THEN
            VT(1)=1D0
         ELSEIF (NA.EQ.2.OR.NA.EQ.4) THEN
            VT(2)=1D0
         ENDIF

      ELSEIF (KR.EQ.10) THEN
         IF(NA.EQ.12) THEN
            VT(1)=HH
            VT(2)=-HH
         ELSEIF (NA.EQ.13) THEN
            VT(2)=1D0
         ELSEIF (NA.EQ.14) THEN
            VT(1)=1D0
         ELSEIF (NA.EQ.23) THEN
```

```
            VT(2)=HH
            VT(3)=-HH
         ELSEIF (NA.EQ.24) THEN
            VT(1)=HH
            VT(3)=-HH
         ELSEIF (NA.EQ.34) THEN
            VT(3)=1D0
         ENDIF

      ELSEIF (KR.EQ.13.OR.KR.EQ.14) THEN
         IF(NA.EQ.12.OR.NA.EQ.14) THEN
            VT(1)=1D0
         ELSEIF (NA.EQ.13.OR.NA.EQ.15) THEN
            VT(2)=1D0
         ELSEIF (NA.EQ.25.OR.NA.EQ.34.OR.NA.EQ.45.OR.NA.EQ.23) THEN
            VT(3)=1D0
         ENDIF

      ELSEIF (KR.EQ.15.OR.KR.EQ.18) THEN
         IF(NA.EQ.13.OR.NA.EQ.23) THEN
            VT(1)=HH
            VT(2)=-HH
         ELSEIF (NA.EQ.14.OR.NA.EQ.24) THEN
            VT(2)=1D0
         ELSEIF (NA.EQ.15.OR.NA.EQ.25) THEN
            VT(1)=1D0
         ELSEIF (NA.EQ.34.OR.NA.EQ.35.OR.NA.EQ.45) THEN
            VT(3)=1D0
         ENDIF

      ELSEIF (KR.EQ.20.OR.KR.EQ.27) THEN
         IF(NA.EQ.46.OR.NA.EQ.45.OR.NA.EQ.35.OR.NA.EQ.36) THEN
            VT(1)=1D0
         ELSEIF (NA.EQ.16.OR.NA.EQ.15.OR.NA.EQ.25.OR.NA.EQ.26) THEN
            VT(2)=1D0
         ELSEIF (NA.EQ.14.OR.NA.EQ.13.OR.NA.EQ.23.OR.NA.EQ.24) THEN
            VT(3)=1D0
         ENDIF

      ENDIF

      RETURN
C----------END VLTANG----------
      END


      SUBROUTINE PROJECT(NPROJ,KR,NAEXPP,X,XP)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(*),XP(*),XPP(3)

      N=NDIM(KR)
      CALL VEQV(N,XPP,X)
      IF(NPROJ.EQ.1) THEN
        CALL PROJ1(KR,NAEXPP,X,XPP)
      ELSEIF (NPROJ.EQ.2) THEN
        CALL PROJ2(KR,NAEXPP,X,XPP)
      ENDIF
      CALL VEQV(N,XP,XPP)
      RETURN
C----------END PROJECT----------
      END


      SUBROUTINE PROJ1(KR,NAEX,X,XP)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (Z=0D0,H=.5D0,T=1D0/3D0)
      DIMENSION X(*),XP(*)

      N=NDIM(KR)
      CALL VEQV(N,XP,X)
```

```
      IF(KR.EQ.3) THEN
         XP(1)=Z
      ELSEIF (KR.EQ.6) THEN
         IF(NAEX.EQ.1) THEN
            XP(1)=H*(X(1)-X(2))
            XP(2)=-XP(1)
         ELSEIF (NAEX.EQ.2) THEN
            XP(1)=Z
         ELSEIF (NAEX.EQ.3) THEN
            XP(2)=Z
         ENDIF

      ELSEIF (KR.EQ.8.OR.KR.EQ.9) THEN
         IF(NAEX.EQ.1.OR.NAEX.EQ.3) THEN
            XP(2)=Z
         ELSEIF (NAEX.EQ.2.OR.NAEX.EQ.4) THEN
            XP(1)=Z
         ENDIF

      ELSEIF (KR.EQ.10) THEN
         IF(NAEX.EQ.1) THEN
            XP(3)=Z
         ELSEIF (NAEX.EQ.2) THEN
            XP(1)=T*(2D0*X(1)-X(2)-X(3))
            XP(2)=T*(2D0*X(2)-X(3)-X(1))
            XP(3)=T*(2D0*X(3)-X(1)-X(2))
         ELSEIF (NAEX.EQ.3) THEN
            XP(1)=Z
         ELSEIF (NAEX.EQ.4) THEN
            XP(2)=Z
         ENDIF

      ELSEIF (KR.EQ.13.OR.KR.EQ.14) THEN
         IF(NAEX.EQ.1) THEN
            XP(3)=Z
         ELSEIF (NAEX.EQ.2.OR.NAEX.EQ.4) THEN
            XP(2)=Z
         ELSEIF (NAEX.EQ.3.OR.NAEX.EQ.5) THEN
            XP(1)=Z
         ENDIF

      ELSEIF (KR.EQ.15.OR.KR.EQ.18) THEN
         IF(NAEX.EQ.1.OR.NAEX.EQ.2) THEN
            XP(3)=Z
         ELSEIF (NAEX.EQ.3) THEN
            XP(1)=H*(X(1)-X(2))
            XP(2)=-XP(1)
         ELSEIF (NAEX.EQ.4) THEN
            XP(1)=Z
         ELSEIF (NAEX.EQ.5) THEN
            XP(2)=Z
         ENDIF

      ELSEIF (KR.EQ.20.OR.KR.EQ.27) THEN
         IF(NAEX.EQ.1.OR.NAEX.EQ.2) THEN
            XP(1)=Z
         ELSEIF (NAEX.EQ.3.OR.NAEX.EQ.4) THEN
            XP(2)=Z
         ELSEIF (NAEX.EQ.5.OR.NAEX.EQ.6) THEN
            XP(3)=Z
         ENDIF

      ENDIF
      RETURN
C----------END PROJ1------------
      END


      SUBROUTINE PROJ2(KR,NA,X,XP)

      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (Z=0D0,H=.5D0,T=1D0/3D0)
      DIMENSION X(*),XP(*)
```

```
      N=NDIM(KR)
      IF(N.LT.3) THEN     !* DIM < 3
         DO 10 I=1,N
10       XP(I)=Z
         RETURN
      ENDIF

      CALL VEQV(N,XP,X)

      IF(KR.EQ.10) THEN
         IF(NA.EQ.12) THEN
            XP(1)=H*(X(1)-X(2))
            XP(2)=-XP(1)
            XP(3)=Z
         ELSEIF (NA.EQ.13) THEN
            XP(1)=Z
            XP(3)=Z
         ELSEIF (NA.EQ.14) THEN
            XP(2)=Z
            XP(3)=Z
         ELSEIF (NA.EQ.23) THEN
            XP(1)=Z
            XP(2)=H*(X(2)-X(3))
            XP(3)=-XP(2)
         ELSEIF (NA.EQ.24) THEN
            XP(1)=H*(X(3)-X(1))
            XP(2)=Z
            XP(3)=-XP(1)
         ELSEIF (NA.EQ.34) THEN
            XP(1)=Z
            XP(2)=Z
         ENDIF

      ELSEIF (KR.EQ.13.OR.KR.EQ.14) THEN
         IF(NA.EQ.12.OR.NA.EQ.14) THEN
            XP(2)=Z
            XP(3)=Z
         ELSEIF (NA.EQ.13.OR.NA.EQ.15) THEN
            XP(1)=Z
            XP(3)=Z
         ELSEIF (NA.EQ.25.OR.NA.EQ.34.OR.NA.EQ.45.OR.NA.EQ.23) THEN
            XP(1)=Z
            XP(2)=Z
         ENDIF

      ELSEIF (KR.EQ.15.OR.KR.EQ.18) THEN
         IF(NA.EQ.13.OR.NA.EQ.23) THEN
            XP(1)=H*(X(1)-X(2))
            XP(2)=-XP(1)
            XP(3)=Z
         ELSEIF (NA.EQ.14.OR.NA.EQ.24) THEN
            XP(1)=Z
            XP(3)=Z
         ELSEIF (NA.EQ.15.OR.NA.EQ.25) THEN
            XP(2)=Z
            XP(3)=Z
         ELSEIF (NA.EQ.34.OR.NA.EQ.35.OR.NA.EQ.45) THEN
            XP(1)=Z
            XP(2)=Z
         ENDIF

      ELSEIF (KR.EQ.20.OR.KR.EQ.27) THEN
         IF(NA.EQ.46.OR.NA.EQ.45.OR.NA.EQ.35.OR.NA.EQ.36) THEN
            XP(2)=Z
            XP(3)=Z
         ELSEIF (NA.EQ.16.OR.NA.EQ.15.OR.NA.EQ.25.OR.NA.EQ.26) THEN
            XP(1)=Z
            XP(3)=Z
         ELSEIF (NA.EQ.14.OR.NA.EQ.13.OR.NA.EQ.23.OR.NA.EQ.24) THEN
            XP(1)=Z
            XP(2)=Z
         ENDIF
```

```
      ENDIF
      RETURN
C----------END PROJ2------------
      END


      SUBROUTINE JACOBI(XKR,S,KR,N,SFMINV,FD)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XKR(3,KR),S(N),SFMINV(N,3),FD(3*KR)
      CALL KRFID(S,FD,KR)
      CALL MMATBT(N,KR,3,FD,XKR,SFMINV)
      RETURN
C----------END JACOBI-----------
      END


      SUBROUTINE GETPERM(PERA,PERM)

      IMPLICIT REAL*8 (A-H,O-Z)

      DIMENSION PERA(6),PERM(3,3)
      K=0
      DO 100 J=1,3
        DO 100 I=1,J
          K=K+1
          PERM(I,J)=PERA(K)
          IF (I.NE.J) PERM(J,I)=PERM(I,J)
  100 CONTINUE
      RETURN
C----------END GETPERM----------
      END


      SUBROUTINE GETPOT(KR,NELC,POT,HEAD)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION NELC(KR),POT(*),HEAD(KR)

      DO 100 I=1,KR
  100 HEAD(I)=POT(NELC(I))
      RETURN
C----------END GETPOT-----------
      END


      SUBROUTINE GETXKR(KR,COOR,NELC,XKR)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION COOR(3,*),NELC(KR),XKR(3,KR)

      DO 100 J=1,KR
        NIC=NELC(J)
        DO 100 I=1,3
  100   XKR(I,J)=COOR(I,NIC)

      RETURN
C----------END GETXKR-----------
      END


      FUNCTION SHAPEH(XL,HEAD,KR)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XL(*),HEAD(*),F(27)
      CALL KRFI(XL,F,KR)
      CALL MMAB(1,KR,1,F,HEAD,SHAPEH)
      RETURN
C----------END SHAPEH-----------
      END
```

```
      SUBROUTINE SHAPEV(VL,VG,XKR,KR)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION VL(*),VG(3),XKR(3,KR),F(27)
      CALL KRFI(VL,F,KR)
      CALL MMABT(1,KR,3,F,XKR,VG)
      RETURN
C----------END SHAPEV----------
      END


      SUBROUTINE SHAPEX(XL,XG,XKR,KR)

      IMPLICIT REAL*8 (A-H,O-Z)

      COMMON /ORIGIN/ XORIG(3)


      DIMENSION XL(3),XG(3),XKR(3,KR),F(27)
      CALL KRFI(XL,F,KR)
      CALL MMABT(1,KR,3,F,XKR,XG)
      DO 10 I=1,3
   10 XG(I)=XG(I)+XORIG(I)
      RETURN
C----------END SHAPEX----------
      END


      FUNCTION NUMAREA(KR)
      IF      (KR.EQ.3)               THEN
        NUMAREA=2
      ELSEIF (KR.EQ.6)                THEN
        NUMAREA=3
      ELSEIF (KR.EQ.8.OR.KR.EQ.9)     THEN
        NUMAREA=4
      ELSEIF (KR.EQ.10)               THEN
        NUMAREA=4
      ELSEIF (KR.EQ.13.OR.KR.EQ.14) THEN
        NUMAREA=5
      ELSEIF (KR.EQ.15.OR.KR.EQ.18) THEN
        NUMAREA=5
      ELSEIF (KR.EQ.20.OR.KR.EQ.27) THEN
        NUMAREA=6
      ELSE
        NUMAREA=0
      ENDIF
      RETURN
C----------END NUMAREA----------
      END


      LOGICAL FUNCTION KRTEST(KR)

C 14-, 18- and 27-node elements not used!
C      KRTEST=(KR.EQ.3.OR.KR.EQ.6.OR.KR.EQ.8.OR.
C     &        KR.EQ.10.OR.KR.EQ. 9.OR.KR.EQ.15.OR.
C     &        KR.EQ.13.OR.KR.EQ.14.OR.
C     &        KR.EQ.18.OR.KR.EQ.20.OR.KR.EQ.27)
      KRTEST=(KR.EQ.3  .OR. KR.EQ.6  .OR.KR.EQ.8.  OR.
     &        KR.EQ.10 .OR. KR.EQ.13 .OR.KR.EQ.15 .OR. KR.EQ.20)
      RETURN
C----------END KRTEST----------
      END


      FUNCTION NDIM(KR)
      IF (KR.LT.4) THEN
          NDIM=1
      ELSEIF (KR.LT.10) THEN
          NDIM=2
```

```
      ELSE
          NDIM=3
      ENDIF
      RETURN
C----------END NDIM-------------
      END


      SUBROUTINE LSTART(KR,X)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(3)

C---- Starting point for SEARCH-----------------------------------------
      IF(KR.EQ.6) THEN
        X(1)=1D0/3D0
        X(2)=1D0/3D0
      ELSEIF (KR.EQ.10) THEN
        X(1)=0.25D0
        X(2)=0.25D0
        X(3)=0.25D0
      ELSEIF (KR.EQ.15 .OR. KR.EQ.18) THEN
        X(1)=1D0/3D0
        X(2)=1D0/3D0
        X(3)=0D0
      ELSE
        X(1)=0D0
        X(2)=0D0
        X(3)=0D0
      ENDIF
C----------END LSTART-----------
      END


      SUBROUTINE KRFI(XL,F,KR)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XL(*),F(*)
C-----------------------------------------------------------------------
C    KRFI CALLS THE SPECIFIED SHAPEFUNCTION TO SET F(XL)
C-----------------------------------------------------------------------
      IF (KR.EQ.10) CALL IF410(XL,F)
      IF (KR.EQ.13) CALL IF513(XL,F)
      IF (KR.EQ.15) CALL IF615(XL,F)
      IF (KR.EQ.18) CALL IF618(XL,F)
      IF (KR.EQ.20) CALL IF820(XL,F)
      IF (KR.EQ.27) CALL IF827(XL,F)
      IF (KR.EQ. 3) CALL IF23(XL,F)
      IF (KR.EQ. 6) CALL IF36(XL,F)
      IF (KR.EQ. 8) CALL IF48(XL,F)
      IF (KR.EQ. 9) CALL IF49(XL,F)

      RETURN
C----------END KRFI-------------
      END


      SUBROUTINE KRFID(XL,FD,KR)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XL(*),FD(*)
C-----------------------------------------------------------------------
C    KRFID CALLS THE DERIVATIVE OF THE SHAPEFUNCTION TO SET FD(XL)
C-----------------------------------------------------------------------
      IF (KR.EQ.10) CALL IFD410(XL,FD)
      IF (KR.EQ.13) CALL IFD513(XL,FD)
      IF (KR.EQ.15) CALL IFD615(XL,FD)
      IF (KR.EQ.18) CALL IFD618(XL,FD)
      IF (KR.EQ.20) CALL IFD820(XL,FD)
      IF (KR.EQ.27) CALL IFD827(XL,FD)
      IF (KR.EQ. 3) CALL IFD23(XL,FD)
      IF (KR.EQ. 6) CALL IFD36(XL,FD)
```

```
      IF (KR.EQ. 8) CALL IFD48(XL,FD)
      IF (KR.EQ. 9) CALL IFD49(XL,FD)

      RETURN
C----------END KRFID-----------
      END


      FUNCTION ICYCLE(IPOINT,INCR,LE)
      ICYCLE=MOD(IPOINT+INCR,LE)
      IF (ICYCLE.LE.0) ICYCLE=ICYCLE+LE
      RETURN
C----------END ICYCLE----------
      END


      SUBROUTINE ISORT(N,IAR,IARORD)

      DIMENSION IAR(N),IARORD(N)

      DO 100 I=1,N
  100 IARORD(I)=IAR(I)

      IPOS=1
  200 CONTINUE
      IF (IPOS.GT.N) RETURN
      IMIN=IPOS
      IAMIN=IARORD(IPOS)
      DO 300 I=IPOS+1,N
        IF (IARORD(I).LT.IAMIN) THEN
          IAMIN=IARORD(I)
          IMIN=I
        ENDIF
  300 CONTINUE
      MIP=IARORD(IPOS)
      IARORD(IPOS)=IAMIN
      IARORD(IMIN)=MIP
      IEQ=0
      DO 400 I=IPOS+1,N
        IF (IARORD(I).EQ.IAMIN) THEN
          IEQ=IEQ+1
          MIP=IARORD(IPOS+IEQ)
          IARORD(IPOS+IEQ)=IAMIN
          IARORD(I)=MIP
        ENDIF
  400 CONTINUE
      IPOS=IPOS+IEQ+1
      GOTO 200
C----------END ISORT----------
      END


      LOGICAL FUNCTION ANDV(N,LOG)
      LOGICAL LOG(*)
      ANDV=LOG(1)
      DO 100 I=2,N
  100 ANDV=(ANDV.AND.LOG(I))
      RETURN
C----------END ANDV----------
      END


      SUBROUTINE IVEQV(N,IV1,IV2)
      DIMENSION IV1(N),IV2(N)

      DO 100 I=1,N
  100 IV1(I)=IV2(I)
      RETURN
C----------END IVEQV----------
      END
```

```
      SUBROUTINE IVINIT(N,IV,IVAL)
      DIMENSION IV(N)

      IF (N.LT.1) RETURN
      DO 100 I=1,N
  100 IV(I)=IVAL
      RETURN
C----------END IVINIT-----------
      END


      SUBROUTINE MINV3(N,A,AV,DET,IER)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N,N),AV(N,N)
C ----------------------------------------
C   Inverse of a matrix A(N,N) for N=1,2 and 3
C ----------------------------------------

C-------  ERROR-PARAMETER  IER   ---------------------------------------
C
C          IER=0   :  CALCULATIONS OKAY
C          IER=60  :  SPACE-DIMENSION NOT OKAY
C          IER=10  :  DETERMINANT=0
C          IER=50  :  DETERMINANT<0
C-------------------------------------------------------------------------

      IER=0
      IF (N.LT.1 .OR. N.GT.3) GOTO 60
      IF    (N.EQ.1) THEN
        DET=A(1,1)
        IF (DET.LT.0) THEN
          IER=50
        ELSEIF (DET.EQ.0) THEN
          GOTO 50
        ENDIF
  10    AV(1,1)=1./A(1,1)
      ELSEIF(N.EQ.2) THEN
        DET=A(1,1)*A(2,2)-A(1,2)*A(2,1)
        IF (DET.LT.0) THEN
          IER=50
        ELSEIF (DET.EQ.0) THEN
          GOTO 50
        ENDIF
  20    DETV=1./DET
        AV(1,1)= A(2,2)*DETV
        AV(1,2)=-A(1,2)*DETV
        AV(2,1)=-A(2,1)*DETV
        AV(2,2)= A(1,1)*DETV
      ELSE IF(N.EQ.3) THEN
        AM11=A(2,2)*A(3,3)-A(2,3)*A(3,2)
        AM12=-(A(2,1)*A(3,3)-A(2,3)*A(3,1))
        AM13=A(2,1)*A(3,2)-A(2,2)*A(3,1)
        AM21=-(A(1,2)*A(3,3)-A(1,3)*A(3,2))
        AM22=A(1,1)*A(3,3)-A(1,3)*A(3,1)
        AM23=-(A(1,1)*A(3,2)-A(1,2)*A(3,1))
        AM31=A(1,2)*A(2,3)-A(1,3)*A(2,2)
        AM32=-(A(1,1)*A(2,3)-A(1,3)*A(2,1))
        AM33=A(1,1)*A(2,2)-A(1,2)*A(2,1)
        DET=A(1,1)*AM11+A(1,2)*AM12+A(1,3)*AM13
        IF (DET.LT.0) THEN
          IER=50
        ELSEIF (DET.EQ.0) THEN
          GOTO 50
        ENDIF
  30    DETV=1./DET
        AV(1,1)=AM11*DETV
        AV(1,2)=AM21*DETV
        AV(1,3)=AM31*DETV
        AV(2,1)=AM12*DETV
        AV(2,2)=AM22*DETV
        AV(2,3)=AM32*DETV
        AV(3,1)=AM13*DETV
        AV(3,2)=AM23*DETV
        AV(3,3)=AM33*DETV
```

```
      ENDIF
      RETURN
   50 IER=10
      RETURN
   60 IER=60
      END


      SUBROUTINE MMAB(IMX,JMX,KMX,A,B,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(IMX,JMX),B(JMX,KMX),C(IMX,KMX)
      INTEGER CO
C     ------------------------------------------------------
C     Multiplication [A]*[B]=[C]: line of [A] by column of [B]
C     ------------------------------------------------------
      DO 30 LI=1,IMX
        DO 20 CO=1,KMX
          SUM=0.
          DO 10 J=1,JMX
   10     SUM=SUM+A(LI,J)*B(J,CO)
   20     C(LI,CO)=SUM
   30 CONTINUE
      RETURN
      END


      SUBROUTINE MMABT(IMX,JMX,KMX,A,B,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(IMX,JMX),B(KMX,JMX),C(IMX,KMX)
C-----------------------------------------------------------------------
C     MATRIX-MULTIPLICATION [A]*[B]TR=[C]   LINE OF [A] BY LINE OF [B]
C-----------------------------------------------------------------------
      DO 30 LIA=1,IMX
        DO 20 LIB=1,KMX
          SUM=0.
          DO 10 J=1,JMX
   10     SUM=SUM+A(LIA,J)*B(LIB,J)
   20     C(LIA,LIB)=SUM
   30 CONTINUE
      RETURN
      END


      SUBROUTINE MMATB(IMX,JMX,KMX,A,B,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(JMX,IMX),B(JMX,KMX),C(IMX,KMX)
      INTEGER COA,COB
C-----------------------------------------------------------------------
C MATRIX-MULTIPLICATION [A]TR*[B]=[C]  COLUMN [A] BY COLUMN [B]
C-----------------------------------------------------------------------
      DO 30 COA=1,IMX
        DO 20 COB=1,KMX
          SUM=0.
          DO 10 J=1,JMX
   10     SUM=SUM+A(J,COA)*B(J,COB)
   20     C(COA,COB)=SUM
   30 CONTINUE
      RETURN
      END


      SUBROUTINE MMATBT(IMX,JMX,KMX,A,B,C)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(JMX,IMX),B(KMX,JMX),C(IMX,KMX)
      INTEGER COA
C-----------------------------------------------------------------------
C    MATRIX-MULTIPLICATION [A]TR*[B]TR=[C]  COLUMN OF [A] BY LINE OF [B]
C-----------------------------------------------------------------------
      DO 30 COA=1,IMX
        DO 20 LIB=1,KMX
          SUM=0.
```

```
            DO 10 J=1,JMX
   10       SUM=SUM+A(J,COA)*B(LIB,J)
   20       C(COA,LIB)=SUM
   30    CONTINUE
         RETURN
         END


         LOGICAL FUNCTION ORV(N,LOG)
         LOGICAL LOG(*)
         ORV=LOG(1)
         DO 100 I=2,N
  100    ORV=(ORV.OR.LOG(I))
         RETURN
C----------END ORV-------------
         END


         SUBROUTINE TRAN(LMX,KMX,A,AT)
         IMPLICIT REAL*8 (A-H,O-Z)
         DIMENSION A(LMX,KMX),AT(KMX,LMX)
C--------------------------------
C     Transpose of [A] to [AT]
C--------------------------------
         DO 10 LI=1,LMX
         DO 10 KO=1,KMX
    10   AT(KO,LI)=A(LI,KO)
         END


         FUNCTION VABS(N,V)
         IMPLICIT REAL*8 (A-H,O-Z)
         DIMENSION V(N)

         ENTRY R8VABS(N,V)

         VQ=0D0
         DO 100 I=1,N
  100    VQ=VQ+V(I)**2
         VABS=SQRT(VQ)
         R8VABS=VABS
         RETURN
C----------END VABS-------------
         END


         SUBROUTINE VDIF(N,V1,V2,VD)
         IMPLICIT REAL*8 (A-H,O-Z)
         DIMENSION V1(N),V2(N),VD(N)

         ENTRY R8VDIF(N,V1,V2,VD)

         DO 100 I=1,N
  100    VD(I)=V1(I)-V2(I)
         RETURN
C----------END VDIF-------------
         END


         SUBROUTINE VEQV(N,V1,V2)
         IMPLICIT REAL*8 (A-H,O-Z)
         DIMENSION V1(N),V2(N)

         ENTRY R8VEQV(N,V1,V2)

         DO 100 I=1,N
  100    V1(I)=V2(I)
         RETURN
C----------END VEQV-------------
         END
```

```
      SUBROUTINE VINIT(N,V,VAL)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION V(N)

      IF (N.LT.1) RETURN
      DO 100 I=1,N
  100 V(I)=VAL
      RETURN
C----------END VINIT------------
      END


      SUBROUTINE VPROJ(N,VDIR,VBAS,VRES)

      IMPLICIT REAL*8 (A-H,O-Z)

      DIMENSION VDIR(N),VBAS(N),VRES(N)

      VDABSQ=0D0
      DO 100 I=1,N
  100 VDABSQ=VDABSQ+VDIR(I)**2

      IF (VDABSQ.GT.0D0) THEN
        VDB=0D0
        DO 200 I=1,N
  200   VDB=VDB+VDIR(I)*VBAS(I)
        DO 300 I=1,N
  300   VRES(I)=VDIR(I)*VDB/VDABSQ
      ELSE
        DO 400 I=1,N
  400   VRES(I)=0D0
      ENDIF
      RETURN
C----------END VPROJ------------
      END


      FUNCTION VSCP(N,V1,V2)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION V1(N),V2(N)

      ENTRY R8VSCP(N,V1,V2)

      VSCP=0D0
      DO 100 I=1,N
  100 VSCP=VSCP+V1(I)*V2(I)
      R8VSCP=VSCP
      RETURN
C----------END VSCP-------------
      END


      SUBROUTINE VSUM(N,V1,V2,VS)
      IMPLICIT REAL*8  (A-H,O-Z)
      DIMENSION V1(N),V2(N),VS(N)

      ENTRY R8VSUM(N,V1,V2,VS)

      DO 100 I=1,N
  100 VS(I)=V1(I)+V2(I)
      RETURN
C----------END VSUM-------------
      END


      SUBROUTINE VUNIT(N,V,VU)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION V(N),VU(N)

      A=VABS(N,V)
```

```
      IF(A.GT.0D0) THEN
         DO 10 I=1,N
10       VU(I)=V(I)/A
      ELSE
         DO 20 I=1,N
20       VU(I)=0D0
      ENDIF
      RETURN
C----------END VUNIT------------
      END


      SUBROUTINE IAUX615(A1,A2,A3,A4,A5,A6,UM,UP,UQ,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(15)
C      ------------------------------------------------
      F(2)=UM*A2
      F(4)=UM*A4
      F(6)=UM*A6
      F(7)=UQ*A1
      F(8)=UQ*A3
      F(9)=UQ*A5
      F(11)=UP*A2
      F(13)=UP*A4
      F(15)=UP*A6
      F(1)=UM*A1-0.5*(F(6)+F(2)+F(7))
      F(3)=UM*A3-0.5*(F(2)+F(4)+F(8))
      F(5)=UM*A5-0.5*(F(4)+F(6)+F(9))
      F(10)=UP*A1-0.5*(F(15)+F(11)+F(7))
      F(12)=UP*A3-0.5*(F(11)+F(13)+F(8))
      F(14)=UP*A5-0.5*(F(13)+F(15)+F(9))
      END


      SUBROUTINE IAUX820(UM,UP,UQ,A1,A2,A3,A4,A5,A6,A7,A8,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(20)
C      ------------------------------------------------------
      F(2)=UM*A2
      F(4)=UM*A4
      F(6)=UM*A6
      F(8)=UM*A8
      F( 9)=UQ*A1
      F(10)=UQ*A3
      F(11)=UQ*A5
      F(12)=UQ*A7
      F(14)=UP*A2
      F(16)=UP*A4
      F(18)=UP*A6
      F(20)=UP*A8
      F(1)=UM*A1-0.5*(F(8)+F(2)+F(9))
      F(3)=UM*A3-0.5*(F(2)+F(4)+F(10))
      F(5)=UM*A5-0.5*(F(4)+F(6)+F(11))
      F(7)=UM*A7-0.5*(F(6)+F(8)+F(12))
      F(13)=UP*A1-0.5*(F(20)+F(14)+F(9))
      F(15)=UP*A3-0.5*(F(14)+F(16)+F(10))
      F(17)=UP*A5-0.5*(F(16)+F(18)+F(11))
      F(19)=UP*A7-0.5*(F(18)+F(20)+F(12))
      END


      SUBROUTINE IF23(XI,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XI(1),F(3)
C  ---------------------------------------------
      S=XI(1)
      F(2)=1.-S*S
      F(1)=0.5*(1.-S)-0.5*F(2)
      F(3)=0.5*(1.+S)-0.5*F(2)
      END
```

```
      SUBROUTINE IF36(XI,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(6),XI(*)
C -----------------------------------------------
      AL=XI(1)
      BL=XI(2)
      CL=1.-AL-BL
      F(2)=4.*AL*BL
      F(4)=4.*BL*CL
      F(6)=4.*CL*AL
      F(1)=AL-0.5*(F(6)+F(2))
      F(3)=BL-0.5*(F(2)+F(4))
      F(5)=CL-0.5*(F(4)+F(6))
      END


      SUBROUTINE IF410(XI,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XI(3),F(10)
C      -----------------------------------------------
      AL=XI(1)
      BL=XI(2)
      CL=XI(3)
      DL=1.-AL-BL-CL
      F(2)=4.*AL*BL
      F(8)=4.*BL*CL
      F(7)=4.*CL*AL
      F(6)=4.*AL*DL
      F(4)=4.*BL*DL
      F(9)=4.*CL*DL
      F(1) =AL-0.5*(F(7)+F(2)+F(6))
      F(3) =BL-0.5*(F(2)+F(8)+F(4))
      F(10)=CL-0.5*(F(8)+F(7)+F(9))
      F(5) =DL-0.5*(F(6)+F(4)+F(9))
      END


      SUBROUTINE IF48(XI,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(8),XI(*)
      FL(S)=0.5*(1.+S)
      FQ(S)=1.-S*S
C  -----------------------------------------------
      S=XI(1)
      T=XI(2)
      F(2)=FQ(S)*FL(-T)
      F(4)=FL(S)*FQ(T)
      F(6)=FQ(S)*FL(T)
      F(8)=FL(-S)*FQ(T)
      F(1)=FL(-S)*FL(-T)-0.5*(F(8)+F(2))
      F(3)=FL( S)*FL(-T)-0.5*(F(2)+F(4))
      F(5)=FL( S)*FL( T)-0.5*(F(4)+F(6))
      F(7)=FL(-S)*FL( T)-0.5*(F(6)+F(8))
      END


      SUBROUTINE IF49(XI,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(9),XI(*)
      FC(S)=0.5*(S*S+S)
      FQ(S)=1.-S*S
C      -----------------------------------------------
      S=XI(1)
      T=XI(2)
      F(1)=FC(-S)*FC(-T)
      F(2)=FQ( S)*FC(-T)
      F(3)=FC( S)*FC(-T)
      F(4)=FC( S)*FQ( T)
      F(5)=FC( S)*FC( T)
```

```
      F(6)=FQ( S)*FC( T)
      F(7)=FC(-S)*FC( T)
      F(8)=FC(-S)*FQ( T)
      F(9)=FQ( S)*FQ( T)
      END


      SUBROUTINE IF513(XI,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XI(3),F(13)
      FL(S)=0.5*(1.+S)
      FQ(S)=1.-S*S
C      ---------------------------------------------
      S=XI(1)
      T=XI(2)
      U=XI(3)
      A1= FL(-S)*FL(-T)
      A3= FL(S)*FL(-T)
      A5= FL(S)*FL(T)
      A7= FL(-S)*FL(T)
      UM=FL(-U)
      UQ=FQ(U)
C      -------------------------
      F(2)=FQ( S)*FL(-T)*UM
      F(4)=FL( S)*FQ( T)*UM
      F(6)=FQ( S)*FL( T)*UM
      F(8)=FL(-S)*FQ( T)*UM
      F(9) =A1*UQ
      F(10)=A3*UQ
      F(11)=A5*UQ
      F(12)=A7*UQ
      F(1) =A1*UM-0.5*(F(8)+F(2)+F(9))
      F(3) =A3*UM-0.5*(F(2)+F(4)+F(10))
      F(5) =A5*UM-0.5*(F(4)+F(6)+F(11))
      F(7) =A7*UM-0.5*(F(6)+F(8)+F(12))
      F(13)=FL(U)-0.5*UQ
      END


      SUBROUTINE IF615(XI,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XI(3),F(15)
C      ---------------------------------------------
      AL=XI(1)
      BL=XI(2)
      CL=1.-AL-BL
      U =XI(3)
      A2=4.*AL*BL
      A4=4.*BL*CL
      A6=4.*CL*AL
      UM=0.5*(1.-U)
      UP=0.5*(1.+U)
      UQ=1.-U*U
      CALL IAUX615(AL,A2,BL,A4,CL,A6,UM,UP,UQ,F)
      END


      SUBROUTINE IF618(XI,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(18),XI(3),FT(6),FU(3)
C      -----------------------------------------------------
      AL=XI(1)
      BL=XI(2)
      CL=1.-AL-BL
      U =XI(3)
      FT(1)=AL*(AL+AL-1.)
      FT(3)=BL*(BL+BL-1.)
      FT(5)=CL*(CL+CL-1.)
      FT(2)=4.*AL*BL
      FT(4)=4.*BL*CL
      FT(6)=4.*CL*AL
      FU(1)=0.5*(U*U-U)
```

```
      FU(2)=1.-U*U
      FU(3)=0.5*(U*U+U)
      N=0
      DO 20 IU=1,3
        A=FU(IU)
        DO 10 IT=1,6
          N=N+1
          F(N)=FT(IT)*A
   10   CONTINUE
   20 CONTINUE
      END


      SUBROUTINE IF820(XI,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XI(3),F(20)
      FL(S)=0.5*(1.+S)
      FQ(S)=1.-S*S
C     ---------------------------------------------
      S=XI(1)
      T=XI(2)
      U=XI(3)
      A2= FQ( S)*FL(-T)
      A4= FL( S)*FQ( T)
      A6= FQ( S)*FL( T)
      A8= FL(-S)*FQ( T)
      A1= FL(-S)*FL(-T)
      A3= FL( S)*FL(-T)
      A5= FL( S)*FL( T)
      A7= FL(-S)*FL( T)
      UM= FL(-U)
      UP= FL(U)
      UQ= FQ(U)
      CALL IAUX820(UM,UP,UQ,A1,A2,A3,A4,A5,A6,A7,A8,F)
      END


      SUBROUTINE IF827(XI,F)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(27),SQ(3),TQ(3),UQ(3),XI(3),NUM(27)
      DATA NUM/1,2,3,8,9,4,7,6,5,10,11,12,17,18,13,16,15,14,
     1       19,20,21,26,27,22,25,24,23/
C     -----------------------------------------------------
      FC(S)=0.5*(S*S+S)
      FQ(S)=1.-S*S

      S=XI(1)
      T=XI(2)
      U=XI(3)
      SQ(1)=FC(-S)
      SQ(2)=FQ(S)
      SQ(3)=FC( S)
      TQ(1)=FC(-T)
      TQ(2)=FQ(T)
      TQ(3)=FC( T)
      UQ(1)=FC(-U)
      UQ(2)=FQ(U)
      UQ(3)=FC( U)
      L=0
      DO 30 IU=1,3
        UQI=UQ(IU)
        DO 20 IT=1,3
          TQI=TQ(IT)
          DO 10 IS=1,3
            L=L+1
            N=NUM(L)
            F(N)=UQI*TQI*SQ(IS)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      END
```

```
      SUBROUTINE IFD23(XI,FD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XI(1),FD(3)
C -----------------------------------------------
      S=XI(1)
      FD(2)=-S-S
      FD(1)=-0.5-0.5*FD(2)
      FD(3)= 0.5-0.5*FD(2)
      END


      SUBROUTINE IFD36(XI,FD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION FD(6,2),XI(*)
C  ---------------------------------------------------
      AL=XI(1)
      BL=XI(2)
      CL=1.-AL-BL
      FD(2,1)= 4.*BL
      FD(4,1)=-4.*BL
      FD(6,1)= 4.*(CL-AL)
      FD(1,1)= 1.-0.5*(FD(6,1)+FD(2,1))
      FD(3,1)= -0.5*(FD(2,1)+FD(4,1))
      FD(5,1)=-1.-0.5*(FD(4,1)+FD(6,1))
C  --------------------------------
      FD(2,2)= 4.*AL
      FD(4,2)= 4.*(CL-BL)
      FD(6,2)=-4.*AL
      FD(1,2)= -0.5*(FD(6,2)+FD(2,2))
      FD(3,2)= 1.-0.5*(FD(2,2)+FD(4,2))
      FD(5,2)=-1.-0.5*(FD(4,2)+FD(6,2))
      END


      SUBROUTINE IFD410(XI,FD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XI(3),FD(10,3)
C      ------------------------------------------------
      AL=XI(1)
      BL=XI(2)
      CL=XI(3)
      DL=1.-AL-BL-CL
C      ------------------------------------------------
      FD(2,1)=4.*BL
      FD(8,1)=0.
      FD(7,1)=4.*CL
      FD(6,1)=4.*(DL-AL)
      FD(4,1)=-4.*BL
      FD(9,1)=-4.*CL
      FD(1,1) = 1.-0.5*(FD(6,1)+FD(2,1)+FD(7,1))
      FD(3,1) = -0.5*(FD(2,1)+FD(4,1)+FD(8,1))
      FD(10,1)= -0.5*(FD(8,1)+FD(7,1)+FD(9,1))
      FD(5,1) =-1.-0.5*(FD(6,1)+FD(4,1)+FD(9,1))
C      ------------------------------------------------
      FD(2,2)=4.*AL
      FD(8,2)=4.*CL
      FD(7,2)=0.
      FD(6,2)=-4.*AL
      FD(4,2)=4.*(DL-BL)
      FD(9,2)=-4.*CL
      FD(1,2) = -0.5*(FD(6,2)+FD(2,2)+FD(7,2))
      FD(3,2) = 1.-0.5*(FD(2,2)+FD(4,2)+FD(8,2))
      FD(10,2)= -0.5*(FD(8,2)+FD(7,2)+FD(9,2))
      FD(5,2) =-1.-0.5*(FD(6,2)+FD(4,2)+FD(9,2))
C      ------------------------------------------------
      FD(2,3)=0.
      FD(8,3)=4.*BL
      FD(7,3)=4.*AL
      FD(6,3)=-4.*AL
      FD(4,3)=-4.*BL
      FD(9,3)=4.*(DL-CL)
      FD(1,3) = -0.5*(FD(6,3)+FD(2,3)+FD(7,3))
```

```
      FD(3,3) = -0.5*(FD(2,3)+FD(4,3)+FD(8,3))
      FD(10,3)= 1.-0.5*(FD(8,3)+FD(7,3)+FD(9,3))
      FD(5,3) =-1.-0.5*(FD(6,3)+FD(4,3)+FD(9,3))
C      ------------------------------------------------------
      END


      SUBROUTINE IFD48(XI,FD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION FD(8,2),XI(*)
      FL(S)=0.5*(1.+S)
      FQ(S)=1.-S*S
      DFQ(S)=-S-S
C      ------------------------------------------------
      S=XI(1)
      T=XI(2)
      FD(2,1)=DFQ(S)*FL(-T)
      FD(4,1)=0.5*FQ(T)
      FD(6,1)=DFQ(S)*FL(T)
      FD(8,1)=-0.5*FQ(T)
      FD(1,1)=-0.5*FL(-T)-0.5*(FD(8,1)+FD(2,1))
      FD(3,1)= 0.5*FL(-T)-0.5*(FD(2,1)+FD(4,1))
      FD(5,1)= 0.5*FL( T)-0.5*(FD(4,1)+FD(6,1))
      FD(7,1)=-0.5*FL( T)-0.5*(FD(6,1)+FD(8,1))
C      ----------------------------------------
      FD(2,2)=FQ(S)*(-0.5)
      FD(4,2)=FL(S)*DFQ(T)
      FD(6,2)=FQ(S)*0.5
      FD(8,2)=FL(-S)*DFQ(T)
      FD(1,2)=-0.5*FL(-S)-0.5*(FD(8,2)+FD(2,2))
      FD(3,2)=-0.5*FL( S)-0.5*(FD(2,2)+FD(4,2))
      FD(5,2)= 0.5*FL( S)-0.5*(FD(4,2)+FD(6,2))
      FD(7,2)= 0.5*FL(-S)-0.5*(FD(6,2)+FD(8,2))
      END


      SUBROUTINE IFD49(XI,FD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION FD(9,2),XI(*)
      FC(S)=0.5*(S*S+S)
      FQ(S)=1.-S*S
C      ------------------------------------------------
      S=XI(1)
      T=XI(2)
      FD(1,1)=(S-0.5)*FC(-T)
      FD(2,1)=(-S-S)*FC(-T)
      FD(3,1)=(S+0.5)*FC(-T)
      FD(4,1)=(S+0.5)*FQ(T)
      FD(5,1)=(S+0.5)*FC(T)
      FD(6,1)=(-S-S)*FC(T)
      FD(7,1)=(S-0.5)*FC(T)
      FD(8,1)=(S-0.5)*FQ(T)
      FD(9,1)=(-S-S)*FQ(T)
C      ---------------------
      FD(1,2)=FC(-S)*(T-0.5)
      FD(2,2)=FQ(S)*(T-0.5)
      FD(3,2)=FC(S)*(T-0.5)
      FD(4,2)=FC(S)*(-T-T)
      FD(5,2)=FC(S)*(T+0.5)
      FD(6,2)=FQ(S)*(T+0.5)
      FD(7,2)=FC(-S)*(T+0.5)
      FD(8,2)=FC(-S)*(-T-T)
      FD(9,2)=FQ(S)*(-T-T)
      END


      SUBROUTINE IFD513(XI,FD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION FD(13,3),XI(3)
      FL(S)=0.5*(1.+S)
      FQ(S)=1.-S*S
C      ------------------------------------------
```

```
      S=XI(1)
      T=XI(2)
      U=XI(3)
      UM=FL(-U)
      UQ=FQ(U)
C      ------------------------------------------
      DSQ=-S-S
      A1=-0.5*FL(-T)
      A3= 0.5*FL(-T)
      A5= 0.5*FL(T)
      A7=-0.5*FL(T)
      FD(2,1)=DSQ*FL(-T)*UM
      FD(4,1)=0.5*FQ(T)*UM
      FD(6,1)=DSQ*FL(T)*UM
      FD(8,1)=-0.5*FQ(T)*UM
      FD(9,1) =A1*UQ
      FD(10,1)=A3*UQ
      FD(11,1)=A5*UQ
      FD(12,1)=A7*UQ
      FD(1,1)=A1*UM-0.5*(FD(8,1)+FD(2,1)+FD(9,1))
      FD(3,1)=A3*UM-0.5*(FD(2,1)+FD(4,1)+FD(10,1))
      FD(5,1)=A5*UM-0.5*(FD(4,1)+FD(6,1)+FD(11,1))
      FD(7,1)=A7*UM-0.5*(FD(6,1)+FD(8,1)+FD(12,1))
      FD(13,1)=0.
C      ------------------------------------------
      DTQ=-T-T
      A1=-0.5*FL(-S)
      A3=-0.5*FL(S)
      A5= 0.5*FL(S)
      A7= 0.5*FL(-S)
      FD(2,2)=-0.5*FQ(S)*UM
      FD(4,2)=FL(S)*DTQ*UM
      FD(6,2)=0.5*FQ(S)*UM
      FD(8,2)=FL(-S)*DTQ*UM
      FD(9,2) =A1*UQ
      FD(10,2)=A3*UQ
      FD(11,2)=A5*UQ
      FD(12,2)=A7*UQ
      FD(1,2)=A1*UM-0.5*(FD(8,2)+FD(2,2)+FD(9,2))
      FD(3,2)=A3*UM-0.5*(FD(2,2)+FD(4,2)+FD(10,2))
      FD(5,2)=A5*UM-0.5*(FD(4,2)+FD(6,2)+FD(11,2))
      FD(7,2)=A7*UM-0.5*(FD(6,2)+FD(8,2)+FD(12,2))
      FD(13,2)=0.
C      ------------------------------------------
      DUQ=-U-U
      A1=FL(-S)*FL(-T)
      A3=FL(S)*FL(-T)
      A5=FL(S)*FL(T)
      A7=FL(-S)*FL(T)
      FD(2,3)=-0.5*FQ(S)*FL(-T)
      FD(4,3)=-0.5*FL(S)*FQ( T)
      FD(6,3)=-0.5*FQ(S)*FL( T)
      FD(8,3)=-0.5*FL(-S)*FQ(T)
      FD(9,3) =A1*DUQ
      FD(10,3)=A3*DUQ
      FD(11,3)=A5*DUQ
      FD(12,3)=A7*DUQ
      FD(1,3)=-0.5*A1-0.5*(FD(8,3)+FD(2,3)+FD(9,3))
      FD(3,3)=-0.5*A3-0.5*(FD(2,3)+FD(4,3)+FD(10,3))
      FD(5,3)=-0.5*A5-0.5*(FD(4,3)+FD(6,3)+FD(11,3))
      FD(7,3)=-0.5*A7-0.5*(FD(6,3)+FD(8,3)+FD(12,3))
C   0.5-0.5*DUQ
      FD(13,3)=0.5+U
      END


      SUBROUTINE IFD615(XI,FD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XI(3),FD(15,3)
C      ------------------------------------------------------
      AL=XI(1)
      BL=XI(2)
      CL=1.-AL-BL
      U=XI(3)
```

```
C DSA2
      A2= 4.*BL
C DSA4
      A4=-4.*BL
C DSA6
      A6= 4.*(CL-AL)
      UM=0.5*(1.-U)
      UP=0.5*(1.+U)
      UQ=1.-U*U
      CALL IAUX615(1.,A2,0.,A4,-1.,A6,UM,UP,UQ,FD(1,1))
C      ------------------------------------------------
C DTA2
      A2=4.*AL
C DTA4
      A4=4.*(CL-BL)
C DTA6
      A6=-4.*AL
      CALL IAUX615(0.,A2,1.,A4,-1.,A6,UM,UP,UQ,FD(1,2))
C      ------------------------------------------------
      A2=4.*AL*BL
      A4=4.*BL*CL
      A6=4.*CL*AL
      DUQ=-U-U
      CALL IAUX615(AL,A2,BL,A4,CL,A6,-0.5,0.5,DUQ,FD(1,3))
C      ------------------------------------------------
      END


      SUBROUTINE IFD618(XI,FD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XI(3),FD(18,3),FT(6),FTS(6),FTT(6),FU(3),FUU(3)
C      ------------------------------------------------------------
      AL=XI(1)
      BL=XI(2)
      CL=1.-AL-BL
      U =XI(3)
      FT(1)=AL*(AL+AL-1.)
      FT(3)=BL*(BL+BL-1.)
      FT(5)=CL*(CL+CL-1.)
      FT(2)=4.*AL*BL
      FT(4)=4.*BL*CL
      FT(6)=4.*CL*AL
      FU(1)=0.5*(U*U-U)
      FU(2)=1.-U*U
      FU(3)=0.5*(U*U+U)

      FTS(1)=4.*AL-1.
      FTS(3)=0.
      FTS(5)=-(4.*CL-1.)
      FTS(2)=4.*BL
      FTS(4)=-4.*BL
      FTS(6)=4.*(CL-AL)
      FTT(1)=0.
      FTT(3)=4.*BL-1.
      FTT(5)=-(4.*CL-1.)
      FTT(2)=4.*AL
      FTT(4)=4.*(CL-BL)
      FTT(6)=-4.*AL
      FUU(1)=U-0.5
      FUU(2)=-U-U
      FUU(3)=U+0.5

      N=0
      DO 20 IU=1,3
        A=FU(IU)
        DA=FUU(IU)
        DO 10 IT=1,6
          N=N+1
          FD(N,1)=FTS(IT)*A
          FD(N,2)=FTT(IT)*A
          FD(N,3)=FT(IT)*DA
   10   CONTINUE
   20 CONTINUE
      END
```

```
      SUBROUTINE IFD820(XI,FD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION FD(20,3),XI(3)
      FL(S)=0.5*(1.+S)
      FQ(S)=1.-S*S
C      ------------------------------------------------
      S=XI(1)
      T=XI(2)
      U=XI(3)
      UM=FL(-U)
      UP=FL(U)
      UQ=FQ(U)
C      ------------------------------------------------
      DSQ=-S-S
      A2=DSQ*FL(-T)
      A4= 0.5*FQ(T)
      A6=DSQ*FL(T)
      A8=-0.5*FQ(T)
      A1=-0.5*FL(-T)
      A3= 0.5*FL(-T)
      A5= 0.5*FL( T)
      A7=-0.5*FL( T)
      CALL IAUX820(UM,UP,UQ,A1,A2,A3,A4,A5,A6,A7,A8,FD(1,1))
C      ------------------------------------------------------
      DTQ=-T-T
      A2=-0.5*FQ(S)
      A4=DTQ*FL(S)
      A6= 0.5*FQ(S)
      A8=DTQ*FL(-S)
      A1=-0.5*FL(-S)
      A3=-0.5*FL( S)
      A5= 0.5*FL( S)
      A7= 0.5*FL(-S)
      CALL IAUX820(UM,UP,UQ,A1,A2,A3,A4,A5,A6,A7,A8,FD(1,2))
C      ------------------------------------------------------
      UM=-0.5
      UP=0.5
      UQ=-U-U
      A2=FQ( S)*FL(-T)
      A4=FL( S)*FQ( T)
      A6=FQ( S)*FL( T)
      A8=FL(-S)*FQ( T)
      A1=FL(-S)*FL(-T)
      A3=FL( S)*FL(-T)
      A5=FL( S)*FL( T)
      A7=FL(-S)*FL( T)
      CALL IAUX820(UM,UP,UQ,A1,A2,A3,A4,A5,A6,A7,A8,FD(1,3))
      END


      SUBROUTINE IFD827(XI,FD)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION FD(27,3),SQ(3),TQ(3),UQ(3),DSQ(3),DTQ(3),DUQ(3)
     +          ,XI(3)
      INTEGER NUM(27)
      DATA NUM/1,2,3,8,9,4,7,6,5,10,11,12,17,18,13,16,15,14,
     1       19,20,21,26,27,22,25,24,23/
C      ------------------------------------------------------
      FC(S)=0.5*(S*S+S)
      FQ(S)=1.-S*S

      S=XI(1)
      T=XI(2)
      U=XI(3)
      SQ(1)=FC(-S)
      SQ(2)=FQ(S)
      SQ(3)=FC( S)
      TQ(1)=FC(-T)
      TQ(2)=FQ(T)
      TQ(3)=FC( T)
      UQ(1)=FC(-U)
      UQ(2)=FQ(U)
      UQ(3)=FC( U)
```

```
      DSQ(1)=S-0.5
      DSQ(2)=-S-S
      DSQ(3)=S+0.5
      DTQ(1)=T-0.5
      DTQ(2)=-T-T
      DTQ(3)=T+0.5
      DUQ(1)=U-0.5
      DUQ(2)=-U-U
      DUQ(3)=U+0.5

      L=0
      DO 30 IU=1,3
        UQI=UQ(IU)
        DUQI=DUQ(IU)
        DO 20 IT=1,3
          TQI=TQ(IT)
          DTQI=DTQ(IT)
          DO 10 IS=1,3
            L=L+1
            N=NUM(L)
            FD(N,1)=UQI*TQI*DSQ(IS)
            FD(N,2)=UQI*DTQI*SQ(IS)
            FD(N,3)=DUQI*TQI*SQ(IS)
   10     CONTINUE
   20   CONTINUE
   30 CONTINUE
      END



      SUBROUTINE CLOSEF(NUNIT)
      LOGICAL LIOS
      CHARACTER FNAME*512,COM*257

      IOS=0
      INQUIRE (NUNIT,IOSTAT=IOS,NAME=FNAME)
      CLOSE (NUNIT)
      IF (IOS.NE.0) RETURN
      CALL BLKOUT(FNAME,LFN)

      CALL SCLVAR('FTN###LN',1,'STRING')
      CALL SCLVAR('FTN###STATUS',1,'STATUS')
      CALL WRTCVAR('FTN###LN',1,LFN,FNAME)
      COM='incl (''IF $file($fname(ftn###ln),ATTACHED) AND'//
     &    ' $file($fname(ftn###ln),P) THEN;'//
     &    'detf $fname(ftn###ln);IFEND'') status=ftn###status'
      CALL LENWOB(COM,LC)
      CALL SCLCMD(COM(:LC))
      CALL REDBVAR('FTN###STATUS.NORMAL',1,LIOS,IBOOL)
      IF (.NOT.LIOS) IOS=100
      CALL DELV('FTN###STATUS')
      CALL DELV('FTN###LN')

      RETURN
C----------END CLOSEF-----------
      END



      SUBROUTINE BLKOUT(STRING,LENGTH)
*  Take out the blanks from the beginning of a string and
*  give the length till last non blank character
*  C.W  14.1.86

      CHARACTER*(*)  STRING
      INTEGER        START,FINISH

      START=1
      FINISH=LEN(STRING)
1     IF (STRING(START:START).EQ.' ') THEN
         START=START+1
         IF (START.LT.FINISH) GO TO 1
      ENDIF
2     IF (STRING(FINISH:FINISH).EQ.' ') THEN
         FINISH=FINISH-1
```

```
         IF (FINISH.GT.START) GO TO 2
      ENDIF

      LENGTH=FINISH-START+1
      STRING=STRING(START:FINISH)

      END


      SUBROUTINE CNTWORD(NUNIT,NBWORD,IEND)
C How much numbers in one line
C C.WACKER 14.2.86

      CHARACTER*133 LINE

      READ(NUNIT,'(A132)',END=900) LINE
      NBWORD=0
      CALL LENWOB(LINE,LS)
C Look for comment at the end of data line: /...
      LASH=INDEX(LINE(1:LS),'/')
      IF (LASH.GT.0) THEN
         CALL LENWOB(LINE(1:LASH-1),LS)
      ENDIF

      LINE(LS+1:LS+1)=' '

      DO 100 I=1,LS
         IF (LINE(I:I).NE.' '.AND.LINE(I:I).NE.',') THEN
            IF (LINE(I+1:I+1).EQ.' '.OR.LINE(I+1:I+1).EQ.',') THEN
               NBWORD=NBWORD+1
            ENDIF
         ENDIF
100   CONTINUE

      BACKSPACE(UNIT=NUNIT)
      IEND=0
      RETURN

900   IEND=-1
      END


      SUBROUTINE CPUTIM(LUN,MESS,USER)

C  Print CPU-time since last call together with supplied message
C  on logical unit LUN (first call : initialize).
C  Send short message to user terminal.

      CHARACTER*(*)  MESS,USER
      CHARACTER*133  MESS1,MESS2,MESS3
      CHARACTER*40   BUFF

      DATA  TIM0, TIM1, TIM2 /0.,0.,0./

       TIM1= TIM2

C The NOS/VE system subroutine returns the CPU time since login
      TIM2=SECOND()
      IF( TIM0.GT.0) GOTO 10
       TIM0= TIM2
      RETURN

10    TIME=( TIM2- TIM1)
      TTIM=( TIM2- TIM0)

      LMESS=MIN(130,LEN(MESS))
      WRITE(MESS1,20001) MESS(:LMESS)
      WRITE(MESS2,20002) TIME
      WRITE(MESS3,20003) TTIM
      CALL LENWOB(MESS1,LMESS1)
      CALL LENWOB(MESS2,LMESS2)
      CALL LENWOB(MESS3,LMESS3)
```

```
C Message to unit LUN
      WRITE(LUN,20000) MESS,TIME,TTIM

C Message to JOB LOG
      CALL REMARK(MESS1(:LMESS1))
      CALL REMARK(MESS2(:LMESS2))
      CALL REMARK(MESS3(:LMESS3))

C Message to $OUTPUT
      CALL TERMOUT(LO)
      WRITE (LO,'(A)') ' '
      WRITE (LO,'(A)') MESS1(:LMESS1)
      WRITE (LO,'(A)') MESS2(:LMESS2)
      WRITE (LO,'(A)') MESS3(:LMESS3)
      CALL FPROMPT(' ')
      CALL OFFTERM(LO)

      RETURN

20000 FORMAT(         /'$ ',A
     &                /'$ TIME USED FOR THIS STEP:',F8.2,' SECONDS'
     &                /'$ TOTAL TIME USED:         ',F8.2,' SECONDS'/)
20001 FORMAT(' $ ',A)
20002 FORMAT(' $ TIME USED FOR THIS STEP:',F8.2,' SECONDS')
20003 FORMAT(' $ TOTAL TIME USED:         ',F8.2,' SECONDS')

      END



      SUBROUTINE DAYTIM(DTM)
C     ****************************************
C     ** VERSION FUER NOS/VE  CH.K.   AUG.86 **
C     ****************************************
C     BENOETIGTE NOS/VE SYSTEMROUTINEN: DATE
C                                       TIME

C  RETURN DATE AND TIME IN CHARACTER STRING DTM.
C  FORMAT: DD-MMM-YY  HH:MM  (BOUNDED TO THE LEFT IN DTM).
C  DTM MUST BE OF LENGTH>=16 FOR FULL DATE AND TIME.

      CHARACTER*3   MONTH(12)
      CHARACTER*10  CDATE
      CHARACTER*8   CTIME
      CHARACTER*(*) DTM
      DATA MONTH /'JAN','FEB','MAR','APR','MAY','JUN',
     1            'JUL','AUG','SEP','OCT','NOV','DEC'/

      CALL DATE(CDATE)
      CALL TIME(CTIME)
      DO 10 I=1,12
        READ(CDATE(6:7),'(I2)')IM
        IF(IM.EQ.I) DTM(4:6)=MONTH(IM)
10    CONTINUE
      DTM(1:3)=CDATE(9:10)//'-'
      DTM(7:16)='-'//CDATE(3:4)//'  '//CTIME(1:5)
      RETURN
      END



      SUBROUTINE FINDKW(NUNIT,STRING,NBLINE)
C     ****************************************
C     ** VERSION FUER NOS/VE  CH.K.   AUG.86 **
C     ****************************************
C Attempt to find the keyword string in file connected to unit=nunit.
C It returns the line number where keyword was found.
C If not found program return with a warning and NBLINE=0
C Keyword is searched in English, but also in French and German
C C.Wacker 11.2.86
      PARAMETER  (MCLE=30)
      CHARACTER  LINE*80,STRING*(*),BIP,FSTRING*20,DSTRING*20
     &          ,STRING2*20
      CHARACTER  FORM*2,FORM1*5
      LOGICAL    NOBIP
```

```
      REWIND(NUNIT)
      ENTRY FWDKW(NUNIT,STRING,NBLINE)
      BACKSPACE(NUNIT)
      BIP=CHAR(7)
      N=0
      NBLINE=0
      NOBIP=(STRING(:1).EQ.'@')
      IF (NOBIP) THEN
        STRING2=(STRING(2:LEN(STRING)))
      ELSE
        STRING2=STRING
      ENDIF

C UPCAS      CALL UPCAS(STRING2,20)
C Allows English, French and German
      IF (STRING.EQ.'COORDINATES') THEN
            FSTRING='COORDONNEES'
            DSTRING='KOORDINATEN'
            LS=11
            LSF=LS
            LSD=LS
      ELSE IF (STRING.EQ.'ELEMENTS') THEN
                  FSTRING='ELEMENTS'
                  DSTRING='ELEMENTE'
                  LS=8
                  LSF=LS
                  LSD=LS
      ELSE IF (STRING.EQ.'COLUMNS') THEN
                  FSTRING='COLONNES'
                  DSTRING='SAEULEN'
                  LS=7
                  LSF=8
                  LSD=LS
      ELSE IF (STRING.EQ.'CUTS') THEN
                  FSTRING='COUPES'
                  DSTRING='SCHNITTE'
                  LS=4
                  LSF=6
                  LSD=8
      ELSE IF (STRING.EQ.'RESULTS') THEN
                  FSTRING='RESULTATS'
                  DSTRING='RESULTATEN'
                  LS=7
                  LSF=9
                  LSD=10
      ELSE IF (STRING.EQ.'PERMEABILITIES') THEN
                  FSTRING='PERMEABILITES'
                  DSTRING='PERMEABILITAETEN'
                  LS=14
                  LSF=13
                  LSD=16
      ELSE IF (STRING.EQ.'NODAL CONDITIONS') THEN
                  FSTRING='CONDITIONS NODALES'
                  DSTRING='KNOTENBEDINGUNG'
                  LS=16
                  LSF=18
                  LSD=15
      ELSE IF (STRING.EQ.'INFILTRATION') THEN
                  FSTRING='ALIMENTATION'
                  DSTRING='INFILTRATION'
                  LS=12
                  LSF=12
                  LSD=12
      ELSE
         LS=LEN(STRING)
         FSTRING=STRING2
         DSTRING=STRING2
         LSF=LS
         LSD=LS
      ENDIF
C Main loop
100   READ(NUNIT,'(A)',IOSTAT=IOS,ERR=900) LINE
C UPCAS900   CALL UPCAS(LINE,80)
900   IF (IOS.EQ.0) THEN
```

```
         N=N+1
         IF (INDEX(LINE(1:MCLE),STRING2(1:LS)).EQ.0.AND.
     &       INDEX(LINE(1:MCLE),FSTRING(1:LSF)).EQ.0.AND.
     &       INDEX(LINE(1:MCLE),DSTRING(1:LSD)).EQ.0) THEN
            GO TO 100
         ELSE
            NBLINE=N
         ENDIF
       ELSE
C No error message if orientation keyword is not found, NL=0,
C or @ is first string character
         IF (STRING2(1:LS).EQ.'ORIENTATION'.OR.NOBIP) THEN
            NBLINE=0
            RETURN
         ENDIF
         IF (STRING2(1:LS).EQ.FSTRING(1:LSF)) THEN
            PRINT *,' Keyword ' ,STRING2(1:LS), ' or ',
     &                 DSTRING(1:LSD), ' not found!', BIP
         ELSE
            PRINT *,' Keyword ' ,STRING2(1:LS), ' or ',
     &                 FSTRING(1:LSF), ' or ' ,DSTRING(1:LSD),
     &                 ' not found!' ,BIP
         ENDIF
         NBLINE=0
         RETURN
       ENDIF
C If keyword is ORIENTATION, nbline is set to his value
       IF (STRING2(1:LS).EQ.'ORIENTATION') THEN
          CALL LENWOB(LINE,LS)
          WRITE(FORM,'(I2)') LS-INDEX(LINE,'=')
          FORM1='(I'//FORM//')'
          READ(LINE(INDEX(LINE,'=')+1:LS),FORM1,ERR=910) NBLINE
       ENDIF
       RETURN
910    PRINT *,'ERROR in orientation number',BIP
       NBLINE=0
       RETURN
       END



       SUBROUTINE LENWOB(STRING,FINISH)
*  LENgth WithOut Blank
*  C.W. 15.1.85

       CHARACTER*(*)  STRING
       INTEGER        FINISH

       FINISH=LEN(STRING)
       IF (FINISH.EQ.1) RETURN

1      IF (STRING(FINISH:FINISH).EQ.' ') THEN
          FINISH=FINISH-1
          IF (FINISH.GT.1) GO TO 1
       ENDIF
       RETURN
C----------END LENWOB-----------
       END



       SUBROUTINE MESSAG(USER,LINE)
       CHARACTER*(*) LINE
       CHARACTER*10 USER
       CALL BLKOUT(LINE,L)
       IF (L.GT.80) L=80
       CALL REMARK(LINE(1:L))
       RETURN
       END
```

```
      SUBROUTINE NEWFILE(NUNIT,STRING,DEFFILE,NAME)
C Open a new file on unit=nunit and check if the file already
C exist. If answer to the string (prompt) is empty then DEFFILE
C is the default name for the file.
C C.Wacker 11.2.86
C adapted  on NOS/VE: 02.06.87 A.v.Kaenel

      CHARACTER*(*) STRING,DEFFILE,NAME
      CHARACTER*256 DUMS1,DUMS2
      CHARACTER     FNAME*512,BIP,ANS,B,LNAME*31
      LOGICAL A

      B=' '
      BIP=CHAR(7)

      CALL TERMIO(L5,L6)
      CALL LENWOB(STRING,LS2)
      CALL LENWOB(DEFFILE,LSD)
      DUMS1=STRING(1:LS2)
      DUMS2=DEFFILE(1:LSD)
100   CONTINUE
      CALL FPROMPT(DUMS1(:LS2)//' [Default is : '//
     &   DUMS2(:LSD)//' ] ')
      READ (L5,'(A)') NAME
      IF (NAME.EQ.' ') THEN
         NAME=DEFFILE
      ENDIF
      CALL CREATE(NAME,FNAME,LNAME)
      IF (LNAME.EQ.' ') GOTO 900
      CALL LENWOB(FNAME,LF)
101   OPEN(UNIT=NUNIT,FILE=FNAME(:LF),STATUS='NEW',IOSTAT=IOS,ERR=900)
      IF (IOS.NE.0) GOTO 900
      CALL TERMOFF(L5,L6)
      RETURN
900   CONTINUE
      A=.FALSE.
      CALL QUEST(' FILE OPENING FAILED.YOU WANT TO EXIT? ',A)
      IF (.NOT.A) GOTO 100
      CALL TERMOFF(L5,L6)
      STOP

C----------END NEWFILE----------
      END


      SUBROUTINE NONBLNK(STRING,NB)
C ------------------------------------------------------------------------
C Get position NB of first non-blank character in given string.
C NB=0 if no non-blank character found.
C ------------------------------------------------------------------------
      CHARACTER*(*) STRING

      LS=LEN(STRING)
      DO 10 NB=1,LS
        IF(STRING(NB:NB).NE.' ') RETURN
10    CONTINUE
      NB=0
      RETURN
      END


      SUBROUTINE OLDFILE(NUNIT,STRING,NAME)
C Open an old file on unit=nunit, filename is given after the
C prompt (string) and return to the main program

      CHARACTER*(*)  NAME,STRING
      CHARACTER      ANS,BIP,LNAME*31,STRING2*256,FNAME*512
      LOGICAL        OK,A

      BIP=CHAR(7)

      CALL TERMIO(L5,L6)
100   CALL FPROMPT(STRING)
```

```
      READ (L5,'(A)') NAME
      IF (NAME(1:3).EQ.' EX') RETURN
      IF (NAME.EQ.' ') GO TO 100

      CALL ATTACH(NAME,FNAME,LNAME)
      IF (LNAME.EQ.' ') GOTO 900
101   OPEN(UNIT=NUNIT,FILE=LNAME,STATUS='OLD',IOSTAT=IOS,ERR=900)
      IF (IOS.NE.0) GOTO 900
      CALL TERMOFF(L5,L6)
      RETURN

  900 A=.FALSE.
      CALL QUEST(' File is not okay;You want to quit ?',A)
      IF (.NOT.A) GOTO 100
      CALL TERMOFF(L5,L6)
      STOP
C----------END OLDFILE----------
      END


      SUBROUTINE PROMPT(TEXT)

      LOGICAL FLUSH
      CHARACTER*(*) TEXT

      ENTRY FPROMPT(TEXT) !*terminal flushing
      FLUSH=.TRUE.
      GOTO 99

      ENTRY SPROMPT(TEXT) !*no terminal flushing
      FLUSH=.FALSE.
      GOTO 99

   99 CONTINUE
      CALL LENWOB(TEXT,LT)
      CALL NVEOUT(TEXT(:LT),FLUSH)
      RETURN
      END


      SUBROUTINE QUEST(STRING,ANS)

C Ask the question string and get answer.
C Returned: LOGICAL ANS=.TRUE. if answer is Y or y
C                      .FALSE.if answer is N or n
C If no answer is given (i.e. <RET> only), the value
C of ANS is unchanged --> Default handling!

      CHARACTER*(*) STRING,STRING2*256
      LOGICAL ANS
      CHARACTER*1   CH,A
      A='Y'
      IF (.NOT.ANS) A='N'
      LS=LEN(STRING)
      STRING2=STRING(1:LS)//' [Y/N: def='//A//'] '
      CALL TERMIO(L5,L6)
      WRITE (L6,'(A)') STRING2(1:LS+14)
      READ (L5,'(A1)') CH
      IF (INDEX('Yy',CH).GT.0) ANS=.TRUE.
      IF (INDEX('Nn',CH).GT.0) ANS=.FALSE.
      CALL TERMOFF(L5,L6)
      RETURN
C----------END QUEST----------
      END


      SUBROUTINE R8HEAD(LUN,H,NDIM)

C Read head values from result file.
C The file must be positioned at the line following
C the keyword RESULTS!
```

```
      IMPLICIT REAL*8 (A-H,O-Z)

      DIMENSION H(*)
      CHARACTER*80 LINE

C Initialize terminal I/O
      CALL TERMOUT(LO)

C Search for first line with readable data
      NBL=0
5     NBL=NBL+1
      READ(LUN,*,ERR=5,END=200) NIC,IDI,HH,QQ
      BACKSPACE (LUN)

C Read data in bulks of 2000 lines because the maximum record
C length is limited to 32K bytes.
C Put head value for node NIC into array element H(NIC+1) to
C prevent error (SUBSCRIPTRANGE) if NIC becomes 0 at a blank line.
      CALL FPROMPT(' Reading heads from result file')
10    READ(LUN,*,ERR=100,END=200) (NIC,IDI,H(NIC+1),QQ,K=1,2000)

C If NIC>0 then 2000 lines have been read. Read next bulk
C starting at the last line read in.
      IF(NIC.GT.0) THEN
         BACKSPACE(LUN)
         CALL FPROMPT(' next 2000 values read')
         GOTO 10
      ENDIF

C If NIC=0 then a reading error occured or a blank line was read in
C (Remark: PRIME stops reading at a blank line!!)
C If the file is ok, then the keyword DEBITS must be present in
C the line after the next line (there are 2 blank lines after data).
100   BACKSPACE(LUN)
      READ(LUN,'(A)') LINE
      IF(INDEX(LINE,'DEBITS')+INDEX(LINE,'FLUX').GT.0) GOTO 300

      CALL PROMPT(' ')
      CALL FPROMPT(' Warning from R8HEAD:  Input stopped at line ***:')
      DO 101 I=1,4
101   BACKSPACE(LUN)
      DO 102 I=1,3
      READ(LUN,'(A)') LINE
      IF(I.EQ.2) THEN
         LINE=' ***'//LINE
      ELSE
         LINE='    '//LINE
      ENDIF
102   CALL FPROMPT(LINE)
      CALL PROMPT(' ')
      GOTO 300

200   CALL PROMPT(' Warning from R8HEAD: keyword "DEBITS" or "FLUX" ')
      CALL FPROMPT(
     &  ' not found at end of result file ! (EOF encountered)')

C Rearrange head values to proper order
300   DO 301 I=1,NDIM-1
301   H(I)=H(I+1)

      CALL OFFTERM(LO)
      RETURN
      END


      SUBROUTINE SAVEHED(LOUT,FIOUT,FILIN,PGM,VERS,DTM,USER)
C -------------------------------------------------------------------
C Write headers of output files.
C -------------------------------------------------------------------

      CHARACTER*(*) FIOUT,FILIN,PGM,VERS,DTM,USER
      CHARACTER*60  FILIN1
      CHARACTER*13  TEXT,TEXT1,TEXT2
      DATA TEXT1,TEXT2 /'Input file:  ','Input files: '/
```

```
      WRITE (LOUT,*) 'File name:    ',FIOUT
      WRITE (LOUT,*)
      CALL NONBLNK(DTM,NS)
      IF(NS.GT.0) WRITE (LOUT,*) 'Date:         ',DTM
      CALL NONBLNK(USER,NS)
      IF(NS.GT.0) WRITE (LOUT,*) 'User:         ',USER
      WRITE (LOUT,*) 'Program:      ',PGM,' - ',VERS

      CALL NONBLNK(FILIN,NS)
      IF(NS.LT.1) THEN
         WRITE (LOUT,*)
         RETURN
      ENDIF

      NB=NS-1+INDEX(FILIN(NS:),' ')
      FILIN1=FILIN(NS:NB)
      CALL NONBLNK(FILIN(NB:),NS)
      IF(NS.GT.0) THEN
         TEXT=TEXT2
      ELSE
         TEXT=TEXT1
      ENDIF
      WRITE (LOUT,*) TEXT//FILIN1

10    IF(NS.LT.1) THEN
         WRITE (LOUT,*)
         RETURN
      ENDIF
      NS=NB-1+NS
      NB=NS-1+INDEX(FILIN(NS:),' ')
      FILIN1=FILIN(NS:NB)
      WRITE (LOUT,*) '              '//FILIN1
      CALL NONBLNK(FILIN(NB:),NS)
      GOTO 10
      END


      SUBROUTINE UPCAS2(WORD)
      CHARACTER*(*) WORD

      LW=LEN(WORD)
      DO 10 I=1,LW
        ICH=ICHAR(WORD(I:I))
        IF (ICH.LE.96.OR.ICH.GE.123) GOTO 10
        WORD(I:I)=CHAR(ICH-32)
10    CONTINUE
      RETURN
      END
```